SINCLAIR QL WORLD

MAY / JUNE 1992

JUNE 1992 £1.95

OF SILVER AND GOLD

AT LAST! The Gold Card Review

Software File
MASTERBASIC 3D TERRAIN

ARE YOU A QL CRIMINAL?
The QL and Data Protection

9 770951 933016 05

SINCLAIR QL WORLD

Editor
Helen Armstrong

Publisher
Mark Kasprowicz

Advertising Manager
Jim Peskett

Creative Director
John Stanley

Designer
Steve Billington

Magazine Services
Pauline Wakeling
Frances Maxwell
Linda Miller

Sinclair QL World,
Published By Arcwind Ltd.
The Coach House,
Tackley, Oxon.
OX5 3AH
Tel: 086 983 677
Fax: 086 983 733
ISSN 026806X

If you have any comments or difficulties please write to the Editor and we will do our best to deal with your problem in the magazine, though we cannot guarantee individual replies.
Back issues are available from the publisher price £2.00 UK, £2.75 Europe. Overseas rates on request.

Subscriptions from:
Subscription Department,
Arcwind Ltd.
The Coach House, Medcroft Road,
Tackley, Oxon OX5 3AH
Tel: 0869 83677 Fax: 0869 83733
UK: £21.00
Europe: £32.00
Rest of World: £38.00
Please include your subscription number with any queries
Reprographic Services by
The Wace Group

Printed and Bound by William Gibbons, Planetary road, Wllenhall, West Midlands

© 1992 Arcwind Ltd, Sinclair QL World is published 12 times a year. All rights reserved. Reproduction in whole or in part without written permission is strictly prohibited. We welcome contributions but can not be responsible for any unsolicited contributions, all material must be supplied with a SAE.

CONTENTS

MAY/JUNE 1992

4 TROUBLESHOOTER • Disks, Bad Disks and Interfaces
11 QL SCENE • New High-memory version Conqueror
13 QL SCENE • Miracle Systems cut Gold Card price!
14 OPEN CHANNEL • Archive Field names
16 SOFTWARE FILE • 3-D Terrain
18 SOFTWARE FILE • MasterBasic
22 OF SILVER AND GOLD • Miracle Systems' Gold Card
25 NEW USER GUIDE • ED LINE_NUMBER, TO EXECUTE
30 ARCHIVE ANSWERS • Tables and Lists
33 ARE YOU A QL CRIMINAL? • Data Protection and the QL
34 DIY TOOLKIT • Fast Search Arrays
38 JUST FOR STARTERS • SuperBasic Tutorial/Part 2
42 A QUESTION OF DOTS • Desktop Publishing/Part 2
44 SYSTEMATIC MACHINE CODE PROGRAMMING • Part 7

Changes afoot!

Since the demise of the 'Fat Golden Boy' of publishing, Bob Maxwell, and his slip into the waves off the Canaries in November last year, it became quite clear that what was regarded as the 'status quo' of publishing, as far as Sinclair QL World, and many other titles, were concerned would not remain the same forever. In the final analysis there were bound to be a few changes.

The management buyout at HHL brought with it restructuring demands that brought them to decide that Sinclair QL World would run more efficiently in the hands of a smaller specialist publisher rather than being tied to their standard print methods.

So it comes, perhaps as no surprise to some, that your magazine has changed hands and now rests with a small but solid publisher located far from the elevated towers of central London. Ironic, is it not, that the location of the magazine is now within ten miles of Maxwell's original premises in Oxford? We hope that that will be the only similarity, however.

What can we promise you? Well, as a we are a small publisher, you will know that Sinclair QL World is much, much more important to its owners than ever before in its history. That's part of the beauty of small operations - we're in tune with people's personal interests., instead of thinking of everything in bulk terms.

It's clear to anyone in the know that there are more QL users than there are QL World readers, so from those who are 'part timers' we'd like to hear what you like to see in these modest pages that would return you to the fold. Perhaps that ought to be called the domain of the ditherers - those who might buy the magazine depending on the detailed contents.

For subscribers - a small but very important word. Your subscription is still absolutely valid, and your copies will arrive until the end of your subscription period. You'll notice that the current cover is dated May/June '92. Basically, this year we shall be publishing one fewer editions than we normally do, to allow us a month to get all the various systems into operation. After that it's back to normal. If you are a subscriber, you'll have the full twelve copies delivered to your door and so your renewal will take place a month later than previously.

We want to maintain a supportive role with both Quanta members and 'sole users', and we want the QL to remain at the forefront of people's minds. That requires your support. We hope that you can give it as much as we give you.

Mark Kasprowicz
Publisher

TROUBLE

A PROBLE

Bryan Davies talks about disks, hard disks, bad disks and interfaces.

Miracle Systems were reported to be selling Gold Cards for £225 and Trump Cards for £75 at the March All Formats computer fair in London (see this month's review on page XX). It is quite likely that the sales of both Trump Cards and Gold Cards have exceeded Miracle's initial expectations and allowed them to obtain better prices by buying larger quantities of components. There is a continuing downward trend in microcomputer hardware prices, although we can't expect it to affect a small market such as the QL one as much as it does some others.

Dilwyn Jones Computing is now supplying version 2.01 of WinBack, the hard disk backup program. This version is a considerable improvement upon the one reviewed several months ago. It incorporates the facility to split large files between floppy disks (up to 95 sections! They can be reassembled when restored to hard disk), support for up to eight physical hard disk drives (for ST QL-emulator users), enforcable additional backups of files (answering my complaint that an aborted backup 'stopped play' because the archive bit was set on files which had been copied), exclusion of specified files from the backup process, a directory rebuild facility to allow directory structure to be recreated when the hard disk is reformatted, better behaviour under the Pointer Environment, and full functionality on the ST with QL emulator. In addition, a fast floppy disk copier routine, a utility to alter dataspace, and a device driver (called BLACK_HOLE) like a null device for fast dumping of output and testing of output routines, are supplied. The upgrade costs just the return postage for users who purchased WinBack prior to 15 February 1992, if they are happy with the manual in a Quill or Perfection document file, or £3 if they want it printed. The full price to new purchasers is still £25.

It is interesting to see how much mileage can be obtained from one subject, and the matter of obtaining extra capacity out of double-density 3.5 in floppy disks continues to provide material for articles. In fact, it was an advertisement in an American magazine, that caught my eye this time. The advert was largely in the form of an article, written by someone stated to be a consultant on flexible disk design, and the gist was that the practice of cutting an extra hole in the edge of DD disks, to make them look like HD disks to disk drives, was doomed - in advance - to failure. As the advert was placed by a floppy disk supplier, and the price (and, presumably, the profit margin) of genuine HD disks is higher than that of DD disks, it is not surprising that the hole-cutting procedure was given the thumbs-down, but the article was well-written and sounded sensible enough. The reasons given for condemning the procedure did not include the obvious one: that particles from the hole-cutting can get onto the disk surface and cause damage. The arguments used were that the magnetic coating is thinner and of higher magnetic coercivity on HD than on DD disks, and the write-head recording currents are also higher on HD disks. The thinner coating with its higher coercivity requires higher current during recording.

Failure rate

The astonishing conclusion of the article, following some tests run with high-quality DD disks, was that the ultimate failure rate was 100% when a 1.44 MB format was used. My own experience, with about 400 DD disks, is a failure rate that is effectively zero, so you can guess that I am rather sceptical about the comments. There are two areas where my own usage of DD disks formatted to 1.44 MB is different from that referred to in the article; one is that no extra holes have to be drilled in DD disks for my drive, because it has jumper settings for 720 KB and 1.44 MB, and apparently takes no notice of the extra hole in the HD disk. That means there is no risk from loose particles, but that wasn't stated as a problem anyway. The other reason may also be linked to the jumper settings; it is possible the drive automatically increases the write-head current when the jumpers are set for 1.44 MB disks. What happens over a long period of time is another matter. Maybe data written with high current to low-coercivity magnetic coating gives errors sooner, but the effect would need to be pronounced to be obvious, and I've not noticed anything over several years. Just to add further confusion, the advert indicated that some manufacturers may be putting DD disk platters into HD cases and selling them as genuine HD disks. Incidentally, the cheapest HD disks offered by this supplier cost about 30 pence each in lots of 200 - wouldn't we be happy to pay this for genuine HD disks?

As a tailpiece to this subject, Gerard Phelan (see section below) wrote to say that Inmac (the commercial PC consumable supplier) quote a clipping level of 80% for their 'best quality' floppies, against 40% for the 'budget' variety. The American article stated that their test DD disks had achieved a 70% level, and the ANSI (American National Standards Institute) requirement was only 45%. Without going into what the 'clipping level' is, it can be taken that a high figure is desirable, but it looks as though even cheap, unbranded disks may achieve the ANSI standard. As noted in the previous paragraph, errors may become more likely as the magnetic coating on the disk ages, and there is sense in suggesting that disks be re-formatted at intervals of, say, one to two years. That is, it is a good idea to copy backed-up files to freshly-formatted disks after this period, then reformat the old disks. This advice is not given solely in relation to DD disks which have been formatted to HD; re-copying and re-formatting at intervals is a sensible procedure with any hard or floppy disks. Having said that, I must admit to not consciously taking this precaution myself, although many of my disks get reformatted every few months, for one reason or another, anyway; there have been no obvious incidents to suggest 'old' back-up copies have become corrupted, with either my QL or PC systems.

The following notes are based on a letter from Gerard Phelan, in what, for him, is a very short dissertation - only three sides. As usual, his comments are wide-ranging, which is why this section is separate from Readers' Letters. He went to the QL show in Münster, Germany and, from what he says, business was quite good there.

He and Tony Firshman (of TF Services) use one-third-height DD Mitsubishi disk drives, as I do, and we have all had trouble getting them to work with the Gold Card. There are detail differences in our thinking

SHOOTER

M S O L V E D

regarding what it takes to get the two units to co-operate, but the basic answer has been to alter the FLP_START and/or FLP_STEP function parameters. Gerard and Tony now say that fitting the version 2.28 rom chip to the GC cures the previous incompatibility, although Gerard does say that booting from floppy is not 100% reliable until the system has been switched on for about 10 minutes. It is good to hear of a cure, rather than a work-around, but the fact that it has been effected through the rom casts more than a little doubt upon the explanation previously expressed for the problem - that the floppy disk driver chip in the GC couldn't handle the Mitsubishi drives. As I pointed out before, it is the one-third height drives which are involved, not the one-quarter-height ones which are now the standard size.

Miracle Systems have now modified the Trump Card manual for use with the Gold Card, by adding a supplement at the front. 3.2 MB HD floppy drives are still proving hard to get, but supplies should have been received by now. Business for Miracle was generally brisk.

As yet, I've not seen Jürgen Falkenburg's hard disk interface, and have no idea of how well it functions, other than Gerard's comment that 'it clearly worked'. As Miracle seem, effectively, to have given up on their HDD for the moment (for reasons of cost, presumably), the performance of the JFC units is definitely of interest. One major advantage of the JFC approach is that standard PC hard disk drives and controller cards can be used, and this may make the overall price lower than the Miracle one. The size of the required hardware is considerable, but so is that of the Miracle HDD box. However, the JFC HDD interface makes use of the memory space normally utilised by the rear rom port on the QL so that, although the actual port is free, you cannot use a rom at the same time as the HDD. Although it was indicated otherwise in the February issue, it appears that the JFC adapter for the Gold Card fits into the 64-way expansion port on the QL, not into the rom port.

Jochen Merz was showing *QD* 4.00 and *The Lonely Joker* (a new game). He also sells *text87 Plus-4* in Germany. A UK distributor for his range may have been announced by now.

Ergon had a ZX Spectrum Emulator for the QL, which should run many Spectrum programs (depending on the level of copy-protection) but is not exactly fast. Perhaps of more significance is their format-conversion program *Open World*, which can produce images in QL Mode 4 or Mode 8 from Amiga IFF- or GIF- and IBM CUT- or TIF-format images. The program was created with the C68 compiler.

On the Minerva scene, TF Services sold all their stock. In addition, someone was apparently caught selling a 'pirate' copy of an old version of Minerva! This is a new development, surely? Pirating of software is known to be commonplace, but now hardware too? As Gerard's letter was dated 24th March - just a week before April Fool's Day - maybe this report should be treated with some caution. Any further comment?

Readers' letters

Complaints have been coming in about non- or late delivery of version Plus-4 of **text87**. The latest letters come from **Peter Lund** of Münster in Germany and **J G Wilkinson** of Southampton. As indicated previously, we were advised that part of the delay was due to the program's author being ill for several weeks after Christmas, and it is understood that last-minute hitches with the changes to the program caused some weeks' delay prior to that. What was, perhaps, more worrying was the lack of any comment from Software87. We have now been assured that, outstanding orders have been fulfilled.

J G Wilkinson has now received Plus-4 but expresses himself quite unhappy with its behaviour. It is reported that there have been some adverse comments appearing on BBSs (bulletin boards) too, but I note that Plus-4 is now into version 2.00, which may incorporate fixes for some of the reported problems. A version of Plus-4 for the Atari ST is under development.

Wordprocessor suppliers have not had a happy time during the first couple of months of 1992. Digital Precision have had problems with telephone calls; the DP answerphone was effectively tied up and inaccessible to callers for about five weeks. Authorities were notified and the problem was (they hope) solved at the end of February. Outstanding enquiries and orders were dealt with by early March. This should mean that **L M Furminger** has now received his copies of the *Success* CP/M emulator and *Professional Publisher Toolbox 2*. The same should apply to the orders of **Wallace Caister** and **B Wray**. DP reports receiving plenty of complimentary letters, and very few critical ones, from purchasers of *Perfection*. (*This has been our experience at QL World, too.*)

There have been complaints about **TK Computerware** from **A P Campbell, C McKay** and **D R Scott**. As the complaints were made some months ago, it is to be hoped they have already been sorted out, but TK were asked to comment on them some time ago and no response has been received. Not dealing more quickly with complaints inevitably encourages rumours to circulate.

The unusually high number of complaints about suppliers during the past few months may cause users to feel the whole QL scene is getting shaky, but it should be made clear that Digital Precision, at least, have called several times, and gone into considerable detail about their telephone problem and customer's complaints, and it is clear that much effort has gone into sorting out a situation well beyond their control and possibly malicious. DP understands that new legislation will be appearing in a few months which will make the tracing of telephone menaces easier.

The plea for assistance in getting ribbons for the **Panasonic KX-1124 printer** produced a response from **Alex Munden**. He believes that only the manufacturer's ribbons are satisfactory; he has experienced the usual problem with after-market ribbons - fraying at the joint, leading to inky fluff getting on the paper. The type of ribbon for which the price is given in the Information box below has an ink reserve that can be activated when print starts to look faded. Alex's letter was printed with a ribbon in this state, and it looked fine.

Harking back to the mention some months ago of **EEC** and **A Ingrey**, the latter's complaint about not receiving a refund for a keyboard and interface (apparently lost somewhere in the shipment chain) was settled by the time the January *QL World* was available, but this information had not reached me. The only remaining problem appears to be with the credit card company that issued a credit against the lost keyboard and interface. The items were subsequently replaced by EEC, so B Ingrey had both the goods and his money back. To sort the matter out, he agreed with EEC that they would bill him again. While that is on the face of it a logical step, it fell foul of the credit card company's operating procedures. Once the company knew the goods (albeit, replacement units) had turned up, they wanted to cancel the

JOCHEN MERZ SOFTWARE

Im stillen Winkel 12 - D-4100 Duisburg 11
Germany - Telephone & Fax 0203 501274

New logo, new software and many price reductions (see*)

QD 4 - Editor using the Pointer Environment, with many new options: GOTO BASIC Procedure/Function, Label, improved parameter string, print, search/replace, and MenuConfig (the same as Config, but menu-driven). £38 Upgrade from V3 **£10**

QSpread - Spreadsheet for the pointer Environment. More than 3200 cells. Completely pointer-driven, up to 3 horizontal and vertical splits at the same time. Formulate-orientated, 22 scientific functions, macro functions, help, scrap, etc. **£49**

NEW! NEW!

QMenu - Menu Extension V3 QMenu is a very easy to use interface with predefined menus (e.g. file select, simple choice boxes, select from lists). These menus may be used by super BASIC, machine code and other languages. File-select, dir-select and list-select all re-written with more options, multi-columns etc. **£12.90**

Update from previous version with new manual **£5**

FiFi - the FileFinder FiFi is easy to use and extremely useful. FiFi scans devices or directory trees and searches files or filenames for strings. Combinations are possible, e.g. String 1 AND string 2 BUT NOT string 3 etc. **£13***

DISA - Intelligent Disassembler. Generates ASCII-output which may be assembled using the GST (Macro) Assembler. **£27**

QDOS Reference Manual - This book is a must for all machine-code programmers. It explains how to use QDOS, all traps and vectors, the Thing System, the HOTKEY Syst.11 and much more. It shows which features work on a QL, an Emulator, how to write compatible for future operating systems. 170 pages. **£27***

QPTR - The Pointer Environment Toolkit. Revised manual which describes how to use the Pointer Interface and the Window Manager for Super BASIC and machine code. Examples on disc, keys, macros and extensions for SuperBASIC. **£27***

FLP/RAM Level 2 now with ATR!!! Replacement EPROM for SuperQboards (V1.17 onwards, with & without mouse) or TrumpCard **(please specify!).** New Manual! Real sub-directories! QL-Emulator compatible! About twice as fast, with improved slave-block-handling! (not on JM-ROMs) **£18**

EASYPTR 11 - Create your own Pointer-Env. menus & sprites, use them in your SuperBASIC or machine-code programs. Supports all the PE facilities, even split application sub-windows and uses the Menu Extension! Many examples ! **£49**

SYSTEM - System Tools for QDOS: real file - attributes are added, e.g. write-only, hidden or for selected users only. Works even over net and on winchester. New TRAP#3. New devices (equiv. to QL-Emu.): named pipes, MEM, NUL. 80 new procs & features, e.g. WSET_FATTR, WSET_FTYPE,WSET,FVERS, ADIR, ASTAT **£27.50**

QDESIGN II - more features, improved and more printer drivers (incl.) laser, QSnap which scans text from screen and puts it into scrap, virtual drawing mode on Emulator and Minerva etc. **£48.** Upgrade from version 1 **£13**

VECEDIT Vector-Font-Editor for QDesign **£19**

NEW: SER MOUSE - Software driver which allows you to connect a serial (e.g.) IBM-style) mouse to one of the SER ports of the QL, which then mimics the QIMI interface. On a 3-button mouse, you have ESC, Wake and Sleep in addition. **£14**

DIAMONDS £11 - BrainSmasher **£12** - Arcanoid **£10** - Firebirds £**10** - SuperGamesPack **£25** - QShang **£14** - QSUP System Utility Pack **£26** - Thing & Eprom Manager **£18.50**

QL-EMULATOR for ATARI ST (260, 520, 1040 and all Mega's, but no STE's, not yet). This hardware, together with software turns any ATARI into a QL. The emulator supports the standard MODE 4 as well as a MODE 4 in much higher resolution (768x280). It supports real subdirectories (complete Level 2 device drivers) on floppy, hard-disk and RAM-disc. The complete QL emulator package includes a lot of utility software and extensions. If you are interested in an emulator, you should write for more information. **£139***

FLOPPY adaptor to connect QL diskdrives to the ST **£9**

VERY SPECIAL OFFER; Harddisk unit for the ATARI ST (without harddisk): case for Syquest changeable drive AND 5.25" or 3.5" drive, PSU, host-adaptor (100% QDOS compatible), cables, driver, thermo-controlled ventilator etc. **£120**

ACSI-Hostadaptor (QSDOS comp.) to connect SCSI harddisks to the ATARI ST **£55**

QDOS on a 68030 (in every Mega ST) running at 32MHz: incredible speed (approx. 8 times faster than a ST, 20 times a QL) together with new E-Driver-Software. Write for more details! Price: 32MHz Version **£600**, 25MHz **£500**

Technical Printer Manuals:

EPSON ESC-P/2 for all EPSON 9/24/48 pin printers. 500 pages **£27**

HP DESKJET for the Deskjet family **£15**

We are the German distributor for Software 87 products (German & English versions available!).

Please add £4.50 for p & p (Europe) or £4.50 for one item and £2.50 for every further item (Overseas). All prices excl. V.A.T. E&OE.

TROUBLESHOOTER

refund, which would have resulted in Mr. Ingrey paying twice. So, be clear about procedures when asking for a refund or replacements. EEC incurred a loss on the deal, as they shipped two sets of units and presumably didn't have insurance coverage for the lost units.

H F Banks wrote again to say his system is now working well. He had found that a faulty keyboard was causing some of his problems. He commented that he resets the QL when he uses different formats of disk (he has DD and HD drives). This is something new to me, but my system has DD and ED, whereas Mr. Banks' problemseemed to relate specifically to HD. Both our systems have a Gold Card. Another user with similar equipment has asked for help: A Ingrey has been plagued with corruption of back-up DD disks (about seven so far) and he is unable to pinpoint the source of the trouble. Unless anyone else has experienced this problem when using the GC with DD drives, it looks as though the finger of suspicion points at the QL itself. One possible solution has been reported: changing the rom chip in the GC to version 2.28. This appears to have been successful on a system which previously could be made to produce corrupted ED disks 'to order'. It is not known whether the apparent bug producing corruption on ED disks has the same or similar effect on HD or DD disks.

INFORMATION

Ribbons for Panasonic KX-P1123/1124
£6.50 (type KXP 145), and for KX-P1180 £5.50 (type KXP 115), plus post, packing and VAT in each case:
Microsol Ltd.
1-4 Christina Street
London EC2A 4PA.
Tel. (071) 739-5607
Fax (071) 739-5223

text87 Plus-4:
Software87
33 Savernake Road
London NW3 2JU.

Minerva:
TF Services
12 Bouverie Place
London W2 1RB.
Tel. (071) 724-9053
Fax (071) 706-2379

WinBack 2.01:
Dilwyn Jones Computing
41 Bro Emrys
Tal-y-Bont
Bangor
Gwynedd LL57 3YT.
Tel. 0248 354023

Gold Card, ED disk drives:
Miracle Systems Ltd.
25 Broughton Way
Osbaldwick
York YO1 3BG.
Tel. 0904 423986

QD 4.00, The Lonely Joker, text87:
Jochen Merz Software
Im stillen Winkel 12
W-4100 Duisburg 11
Germany.
Tel/fax (from UK) (01049) 203 501274

QL-HDD interface:
Jürgen Falkenburg (JFC)
Thanweg 36
D-7539 Ersingen
Germany.

ZX Spectrum Emulator, Open World:
Ergon Development
Davide Santachiara
Via Emilio De Marchi 2
I 42100 Reggio Emilia
Italy

QL SCENE

New version of Conqueror, Perfection

Digital Precision have released a completely new version of *PC Conqueror* - the Gold Special Edition - specially designed for use with the Miracle Gold Card and other QL hardware giving more than 1.1 megabytes of ram. With all the features of the standard PC Conqueror, the Gold Special Edition also has:

Full support for expanded memory to the LIM 4.0 standard for PC software without need for separate drivers. On the Gold Card, for instance, users will get 736K of base memory, plus expanded memory variable between 0K and 944K. Expanded memory improves the performace of most PC software and is essential for some; it can also be used for ramdrives.

Full support for HD disks for HD drive users, as well as single- and double-density disks, giving 2880 sectors per disk.

With ED drives, you can now create any number of pseudo-hard disks (with over 3.2 MB per disk) and boot automatically from them. Disk access in this way is typically five times faster than for DD drives.

Automatic sensing and switching between disk formats at DOS level, with manual override.

-Up to just over 50% faster operation than all previous Conquerors on most Gold Cards.

-Screen colours user-configurable.

-Many other features.

-Fully compatible with MS-DOS, DR-DOS and PC-DOS versions 1.0 to 6.0.

The Gold Special Edition package includes the Standard version for users with backup QL systems.

PC Conqueror Gold Special Edition costs £99.95 including Vat and carriage. *Standard Conqueror* has been reduced to £59.95, and *Solution PC Emulator* to £29.95. Existing owners can upgrade for the difference in the current advertised price plus £10.

A year on from Perfection's first full release, we now have Perfection Special Edition. Some examples of the SE's additional features are:

- Storing and recalling up to six 'environment' (complete paragraph style) settings at a keystroke.
- Four more strips to support extra printer functions.
- 64 extra printer translates of up to 63 lines for special printing needs, plus a new print terminate option.
- Printer driver now supports all Perfection string editing commands.
- Total compatibility with all document files (own, previous Perfection, Quill_doc, Editor and SE, Ascii, Psion Four and text87 export, non-Ascii, machine code, compressed data, etc.) and printer data files (own, previous Perfection, Quill printer_dat) - no re-entering.
- Reformatting speed typically four times faster than standard Perfection. Ditto global search and replace. Other speed enhancements, too.
- Enhanced Configurator, Dictionary Utility and Stripsort.
- Greatly upgraded piping of output to Professional Publisher.
- Merging works for both Perfection and non-Perfection files.

Perfection SE costs £99.95, or £139.95 with Spellchecker and three dictionaries. Existing users can upgrade on the same terms as for Conqueror (above) - return disks but not the manuals.

DP are also doing further deals and upgrades, including a Lightning Special Edition Gold Card version. More details next month; see also DP's advertising in this issue.

New bulletin board homes in on QL

John Nicholson in Ramsgate is the sysop of a - fairly - new bulletin board concerned with matters QL. The board, on **043 51722**, is a 24-hour, seven-day board operating at 300/2400 baud, 8 bits (1 stop, no parity). There is a Quanta section devoted mainly to the locl subgroup, in which John is active, as well as sales, jobs ('not a lot in there at the moment') and Comtalk for the airing of computer views. The bbs - Paramount - also says that there is also nearly 40 mB of program files to download.

Once you have logged on and filled out the bbs questionnaire, you receive 30 minutes of long-on time. If you want more time, there is an annual fee of £15 (private) or £25 (business), with a 10% discount to Quanta members. Benefits incurred include unlimited downloads, route planning, discounts on hard and software (no details here), ability to upload password-protected files for secure storage, and the ability to store private personal data at a remote site; and others. With apologies to Tandata users, the bbs modem does not handle 1200/75.

Apart from the given number, John Nicholson can be contacted by on 043 593995.

Technical Review returns

Now out, after a long wait, is *QL Technical Review* issue 7 from Gwm Gwen Hall. Richard Alexander has been suffering various upheavals in his personal life, but now feels confident that his life and CGH are heading for a more settled period. Consequently, although Richard says that the contents of QLTR7 have been "on file for some time", it is all solid stuff as usual, including a (recent) assessment of Perfection by Rich Mellor, and two initial reports on the Miracle Gold Card (for a full review see page XX in this issue of QL World).

The **editing** of QLTR is being taken over by **Bruce Nicholls at 57 Shaftesbury Road, Romford, Essex RM1 2QJ**. CGH will continue to publish.

Other features in QLTR7 include reviews of *Basic Reporter*, *QMenu*, *QL Genealogist* (by John Shaw again!) and *DataDesign* 2.0, with networking hints from Tony Firshman.

CGH Services have also rationalised their Public Domain Library service. CGH have dropped microdrive service altogether - for reasons of simple time economics - and standardised on disk at a standard charge of £2 per disk (to include software, the disk itself and post and packing). Dave Walker's *C68 Compiler* is one of the programs available on a PD basis from CGH.

QL Technical Review costs £1.50 from **CGH Services, Cwm Gwen Hall, Pencader, Dyfed, Cymru SA399HA. Tel. 055934574.** UK post and packing inclusive. Add 10% of total for Europe and 20% for rest of world. For further information/price lists please send an SAE to CGH.

QL SCENE

Quanta upon Tyne

The Tyneside Quanta workshop will take place on Sunday 2 June 1992 at the Moat House Hotel, Newcastle upon Tyne (why have we been hearing complaints about 'no Quanta meetings in the North' recently?). The Quanta North East subgroup have negotiated a conference rate of £20 bed and breakfast or £29 BB and evening meal (half board). The hotel is close to road, rail and airport travel facilities.

The local monthly meeting of the local group will take place on Saturday 27 June at 2pm at Teresa Church, Heaton Road, Newcastle Upon Tyne. Anyone who wants to attend that meeting should contact the Group in advance.

More information from **Derek Stewart at 20 Emily Street, Gateshead, tel. 091 477 5472 or Dennis Crowe at 21 Midhurst Road, Newcastle Upon Tyne, tel. 091 266 5175.** The group can also be contacted via its bbs, 19.30 to 16.00 (it says here), seven days, V21, V22, V22bis, V23, VT100/VT52. Tel. 091 477 5472 (as for Derek Stewart above).

Text 87 OK

Text87 Plus4, the new version of text87, is now reported to have the bugs which dogged its release in March of this year eradicated. "Version 4 is now totally bug-free", says developer Fred Toussi. Plus4 is a complete rewrite of text87, which generated some concern among users by running several months late over the Christmas period. However, orders were fulfilled in March. Some bugs were reported, and have now been ironed out. **Software87 is at 33 Savernake Road, London NW3 2JV.** See their advertisement on page XX of this issue for details.

Miracle Systems cuts Gold Card price

Miracle Systems have reduced the price of their Gold Card memory expansion for the QL (see the review on page XX of this issue). The Gold Card now costs £225 inclusive of VAT, post and packing, with a two0eyar warrantry and 14-day money-back guarantee in line with Miracle's normal custom. The export price is now £200. Miracle will no longer be accepting part-exchange of older products for the Gold Card; this offer ends with the price reduction.

Miracle now has news of delivery times for the awaiting 3.2 MB disk drives, delayed since before Christmas by unforseen price negotiations. "We don't take any money until the goods are sent out," said Miracle's Stewart Honeyball, "But we felt we had a duty to supply drives, as some people had bought Gold Cards on the assumption that they would be able to buy disk drives later." The pricing problem is now apparently sorted out, and the drives should be out by the time you read this.

For more information, consult Miracle's advertisement in this issue, or contact **Miracle Systems at 25 Broughton Way, York, YO1 3BG, UK. Tel. 0904 423986.**

DIY TOOLKIT UPDATE

New programs have been added to current **DIY Toolkit** volumes as part of a continuous policy of updating. The **QLipboard** and screen-reading routines in Volume S are joined by FOUNTFIX_TASK, which locates copies of the CST Thor character set used in programs like **The Editor**, **Turboquill+** and **Perfection**, replacing them with the QL equivalent.

Volume C now includes routines to centre text horizontally and vertically in any window, CSIZE and display MODE. These extensions were suggested by Dr Dario Leslie. Volume E gains a special version of EDLINES, contributed by Minerva author Laurence Reeves, which ensures compatibility with a change in some Minerva roms.

DIY Toolkit volumes cost £3 each plus £4 per order for media and processing. Fixed bundles are available at £20 per six disks plus £4 p&p. Mdv users send one formatted microcassette for each volume. Order from **DIY Toolkit, Cwm Gwen Hall, Pencader, Dyfed, Cymru SA39 9HA. Tel. 0559 34574.**

Figures are better, say word processor publishers

Software publishers Digital Precision and Software 87 have come back to us with comments on Mike Edwards' comparative user report on wordprocessors. Digital Precision comment that Mike's figures - particularly for block moves - for all three programs are affected by the different generations of program being used and his variable experience with them. All three programs have different modes which can be employed. For instance, using The Editor in non-document mode allows it to move faster than in document mode, "which was bolted on afterwards for people who are used to wordprocessors"; the block move used by Mike in The Editor was apparently 'automated' (using a macro, which is possible for advanced users in The Editor and Perfection), but the block move used for Perfection was not; Perfection has a slow option for some processes to allow easy interrupt by the user, especially the novice user. If the tests were rationalised to compare like with like, text87 (3.00) would improve in ratio terms by about 3.5 times and Perfection by about 2.5 x 2.5 x 3.5, ie over 20 times. If text87 4.00 [which was shipped in March - see this page] were used I'd estimate a further 1.5 times, and if Perfection 3.00 was used, a further 1.5 improvement on Perfection 2.00. If Perfection SE was used, that would become about 2.2 times faster. "Perfection can move 80% of 40% of the Bible in about 3 seconds," Freddy Vaccha reminds me. But where to? And just when God thought we were all safely occupied with worrying about the Big Bang, too.

Fred Toussi of Software87 adds: "The reviewer complains that text87 does not allow the user to save his or her selections of commonly-used typefaces, making it difficult to select different typefaces. This has been available in text87 3.00 since early 1990 and is listed on page 15 of the V3.00 manual." Text87 4.00 is now on the market.

OPEN CHANNEL

Open Channel is where you have the opportunity to voice your opinions in *Sinclair QL World*. Whether you want to ask for help with a technical problem, provide somebody with the answer, or just sound off about something which bothers you, write to:
Open Channel, Sinclair QL World, The Coach House, Medcroft Road, Tackley, Oxon OX5 3AH.

FIELD NAMES

I'm a regular user of *Archive*, and I have written several routines in an attempt to keep up with my business clients and accounts. Having twice in the past lost large amounts of data, once due to a power failure and once through a dodgy disk drive (yes, I do keep backup copies, but on both occasions the backups were corrupted as well!), I have come to know the merits of both Chas Dillon's Recover utility and Digital Precision's *Super Media Manager*. However, to get any sense from Recover, it is important to know both the database field names, and the order in which they were created.

The routine attached to this letter is a stand-alone procedure and uses the numfld() and fieldn() functions to list automatically all the field names for a given file. Once loaded, the direct command 'fields' sets the whole thing in motion, and it's then an easy matte to get hard copy of any specified file. Not only is this a useful precaution for recovery purposes but it also has uses in general housekeeping, and in designing screen layouts, especially where a file has more fields than can be listed on screen. I now keep listings of the fields in all my files for safety. This may be useful to other users.

Alex Munden
London N16

PS Good luck with your new owners - I hope they don't list yachting as a hobby!

```
proc fields
cls
input "File name? "temp$
look temp$
lprint tab 5;"File name:
";temp$;"_dbf"
lprint tab 5;"No"; tab 10;"Field
Name"
let x=0
while x<numfld()
lprint tab 5;x; tab 10;fieldn(x)
let x=x+1
endwhile
close
lprint
endproc
```

Editor's comment: No. Only windsurfing. We've had enough of multi-millionaires for the time being.

Editor's notebook

Ah! Back again. Have you ever felt just like a lost file floating about in a memory limbo? That's how I've been feeling for the past month while we (too many of us to list) tried to decide the who, what, why, where and when of the next issue of QL World. We apologise for all the uncertainty (which, fortunately, did not catch up with our readers until the May issue became overdue) but we are getting over that now.

There seem to have been some - ahem - rumours circulating. No, QL World has not gone out of business. That one seems to have originated from an under-informed telephone-answering person at the old Maxwell office at Panini House. Enquiries should have been diverted to Arcwind or to HHL at Greater London House. Yes, your subscriptions are still up and running. See Noticeboard (page 41) for more information.

Because of the delayed schedule, I apologise to authors and dealers where information has been delayed. We're catching up as fast as we can.

It's nice to be Restored again!

QL SETS

Regarding the letter from Dane Kurth in Switzerland, the following will, I hope, be of interest.

There are two character sets in the QL, and these are located in the rom. The precise location will thus be fixed, and will depend on the rom version. These character sets are common to all and any windows, and are the same regardless of whether the window is a Screen or a Console.

The addresses of the character sets on a QL may be found by first finding the start of the channel table SV_CHBAS. This address is contained in the table of System Variables starting at 163840 (SV_IDENT) and, irrespective of the rom type, is always given at address 163960 and provides a long word pointer to the start address of the channel table. Please note that all addresses and displacements are given in decimal. The last entry in the channel table is always given by SV_CHTOP at located 163964. This is also located in the table of system variables.

The channel table consists of either long word pointers to an area in the common heap, which contains the channel definition blocks, or $FF000000 if the channel is closed.

These channel definition blocks, provided they define a screen or a window, in turn contain pointers to the addresses of the two character sets. Thus the start of the channel table irrespective of the rom type will be given by SuperBasic lines such as lines 20 and 40 of the accompanying program.

This start address is always contained in ram, it points to the first of the channel definition blocks and will vary with different roms. It will also vary with the addition of any add-ons. Lines 50 and 60 will provide the start address of the first channel definition block.

Each channel definition block gives at its start address the length of the block. For a console or screen, this will normally be 256 bytes. Line 70 will provide the block length:

Each block has at +10 the channel tag, and from this word it can be determined if the channel is the COMMAND or SYSTEM window ([hash]1 SuperBasic) or the LIST window ([hash]2 SuperBasic). the COMMAND or SYSTEM window is always the first defined block and should never be closed. Lines 80 and 90 will check whether the definition block is required. The return should be 0, corresponding to the COMMAND window.

Once it has been determined that the channel definition block is the one that is required, the address of the two character sets can be found by adding 42 and 46 to the start of the correct channel definition block as in lines 100, 110, 120 and 130.

The returns from lines 120 and 130 should give addresses in the system rom for the two character sets, and should be the same for all consoles or screens in that particular type of rom.

The returned addresses may be checked for validity by PEEKing the first two bytes of each character set. The first set, at the lowest address, should return 31 - the first valid character - and 96 - the number of valid characters in the first set, minus 1. The second set, at the highest of the two addresses, should return 127 - the first valid character of this set - and 64 - the number of valid characters in the second set, minus 1. Lines 140, 150, 160, 170 will provide this.

The addition of lines 10, 20 and 30 to the program provides a small SuperBasic program to provide the character set addresses for any version of the QL.

P Hutley
Dewsbury
West Yorks

```
10 CLS
20 ChTab_add = PEEK_L(163960)
30 PRINT "QL Version = ";VER$
40 PRINT "Ch Tab address  =
    ";ChTab_add
50 PDB_add =
    PEEK_L(ChTab_add)
60 PRINT "Phys Def Blk add =
    ";PDB_add
70 PRINT "P/D Block Length =
    ";PEEK_L(PDB_add)
80 Chk_add = PDB_add + 16
90 PRINT "Channel Tag =
    ";PEEK_W(Chk_add)
100 ChSet2_add = PDB_add + 42
110 ChSet2_add = PDB_add + 46
120 PRINT "Char Set 1 add  =
    ";PEEK_L(ChSet1_add)
130 PRINT "Char Set 2 add  =
    ";PEEK_L(ChSet2_add)
140 PRINT "Check byte CS1/1 =
    ";PEEK(PEEK_L
    (ChSet1_add))
150 PRINT "Check byte CS1/2 =
    ";PEEK(PEEK_L
    (ChSet1_add)+1)
160 PRINT "Check byte CS2/1 =
    ";PEEK(PEEK_L
    (ChSet2_add))
170 PRINT "Check byte CS2/2 =
    ";PEEK(PEEK_L
```

FLOATING

Reading Mr. Clase's letter (February 1992) took me back to the days when I was yet a person, and had not been reduced to a mere row of digits in a computer in Newcastle. Time was when the programs I wrote were used by others beside myself, and their inputs were required to be protected against error.

I enclose a listing of my recollection of the routine that I wrote then for floating point entry, adapted to QL Basic. I have chosen to return the input string rather than the coerced number. There is sometimes profit in keeping a record of one's own representation of numbers considering how Qdos mangles their format.

I have added a facility for the use of the comma as alternative to the period to mark the decimal point. If this is not needed, then the comma may be removed from the parser string, and Line 1090 modified accordingly.

Leading and trailing blanks are catered for. Embedded blanks are prohibited – they do not necessarily cause crashes, but will always result in an incorrect number being stored in the machine – unless they occur immediately after the E. I have allowed for one blank there. Should this not be needed, then line 1070 and 1110 may be simplified.

Overflow protection is provided, if a little crudely. Qdos, at any rate as implemented on Minerva, will correctly cope with a mantissa of any length that fits within the input buffer. Line 1130, statement 3, is intended to guard against problems with this.

The listing has two sections: a test loop and the function proper. It should work ok. I have printed it straight from the file that has just been LRUNned.

I cannot end without a word to say how relieved I am that you survive. Could not do without you, neither I could, my lifeline.

P H Tanner
Glasgow

CHEWED

Four years ago I purchased a dual disk drive. Six months ago the cat chewed through the power supply lead and the second drive was damaged. Only one of the drives now works, on a bench power supply at the repairers. Unfortunately I have been unable to get hold of a replacement power supply. I have been informed by someone at Miracle Systems that the drive needs a power supply tht gives +12V and +5V. The old power supply was on a plug with a lead to the disk drive. The drives were 3.5 in one-third height NEC drives (1036). Can anyone tell me

```
100  prompt$="Enter a Number : ":REPeat testnum
110  num=nummer$(prompt$):PRINT num:IF num=0:EXIT
     testnum
120  END REPeat testnum

1000 DEFine FuNction nummer$(p$)
1010 LOCAL a,b,d,e,k,n,p,s,n$:REPeat nummer
1020 PRINT p$;:INPUT n$:a=0:b=0:d=0:e=0:n=-
     1:s=0:n$&FILL$("0",LEN(n$)=0)
1030 FOR k=1 TO LEN(n$)
1040 p=n$(k) INSTR " +-.0123456789Ee,":a=a__(-(a=0 AND
     p>1)&&k):SELect ON p
1050 =0        :n=0
1060 =1                                              :b=k
1070 =2,3      :n=n&&-(k=a OR k=e+1+(b=e+1))          :s=k
1090 =4,17     :n=n&&-(d=0 AND e=0)                  :d=k
     :n$(k)="."
1100 =15,16    :n=n&&-(a<>k AND b=0 AND e=0)         :e=k
1110 =REMAINDER:n=-(b<a OR b=e+1)&&k
1120 END SELect:IF n=0:EXIT k
1130 END FOR k:IF n>e AND e>0:n=n&&-(ABS(n$(e+1
     TO))<(618-e)):k=e+1+(s>e)
1140 IF n>e AND n>s:RETurn n$:ELSE PRINT TO LEN(p$)+k-
     1;CHR$(190);" Error"
1160 END REPeat nummer:END DEFine
```

where I can get hold of a power supply suitable for the drive, and a replacement second drive, or a way of using the case and the one drive that still works? I know nothing about electronics.

M J Williams
Telford
Shropshire

DISK LOSS

First, congratulations on the removal of the dreaded name from the front cover. It must have been a worrying time for you all. And needless to say, for all of us out here.

In the December 1991 issue of *QL World* there is a letter from Mr J Bissonette, regarding loss of files from floppy disks, for no apparent reason. In the current issue, April issue, Bryan Davies speaks of it, without any definite conclusion. No mention is made of make or size of disk or disk drive.

I have recently suffered from exactly the same trouble, so I can fill in exact details.

About two years ago I bought my first disk drive at a radio rally. It was a Quest single-sided 5.25 in Shugart drive. This was connected to my QL via a CST disk interface.

It worked wonderfully. The disks were first Parrot, and later Select DSDD, but one side was unused. In an old copy of Electronics and Computing, an article described how to punch an extra index hole on the 'other' side, and a write-defeat slot on the edge. This I did, without any trouble, about two years ago, without any problems until last December when the dreaded Bad Medium appeared.

The disks on which this occurred were the earliest ones saved, and some (not all) would not load. A month or so later, I found that more and more files were becoming bad. The directory showed them as present, but later a DIR produced Not Found. The make of disk, or which side was in used, did not make any difference. The process seems to have be accelerating. I have today been forced to reformat and save a Boot program saved only two months ago.

I must emphasise that the problem has nothing whatever to do with handling care, magnetic fields or storage. I am very fussy about these. I have added two 3.5 in drives about 18 months ago, and these have been impeccable. Why am I using the 5.25 in ones? I shall of course discontinue them now.

My conclusions are (1) that the drive is not producing enough magnetic saturation. I do not think it is due to the disks. I have dismantled the drive and cleaned the head, without any cure. For the record, I am a retired electronics engineer.

R Snow
Bath
Avon

SOFTWARE FILE

3D TERRAIN

INFORMATION
Program: *3D Terrain* V1 . 0
Price: Check with supplier
Supplier: Sharp's, Box 326, Mechanicsville, Virginia 23111, USA.
Tel. (USA) - (804) - 730 - 9697
Fax. (USA) - (804) - 746 - 1978

3D Terrain throws a 'net' over detailed landscape shapes which can be given colour and form. It works well, says Bryan Davies.

Purely as an accident of timing, this is the first program I've reviewed since having a *Gold Card* available. It's a program which does a great deal of drawing on the screen. Both preliminary calculations and drawing on the screen are rather slow on the basic QL (not just for this program), so this proved a good opportunity to see what improvement the GC gives.

3D Terrain is a drawing program which produces 'terrain-like' pictures on the screen. If you have seen screen shapes apparently moulded in wire-mesh chicken netting, you will know what this means. The external surfaces of mountains; valleys etc., are displayed as though covered by a layer of netting, giving an appearance similar to contour lines on a map. This mapping technique is commonly used nowadays, for instance in the simulation of the effects proposed building work will have on an existing landscape. Typically, a photographic image of the existing area is overlaid by mesh representing the shape of the new development, so that the visual impact of individual features of the new buildings can be assessed. The program reviewed makes no claims to utilitarian purposes, however; it is simply a means of displaying interesting shapes, and modifying them.

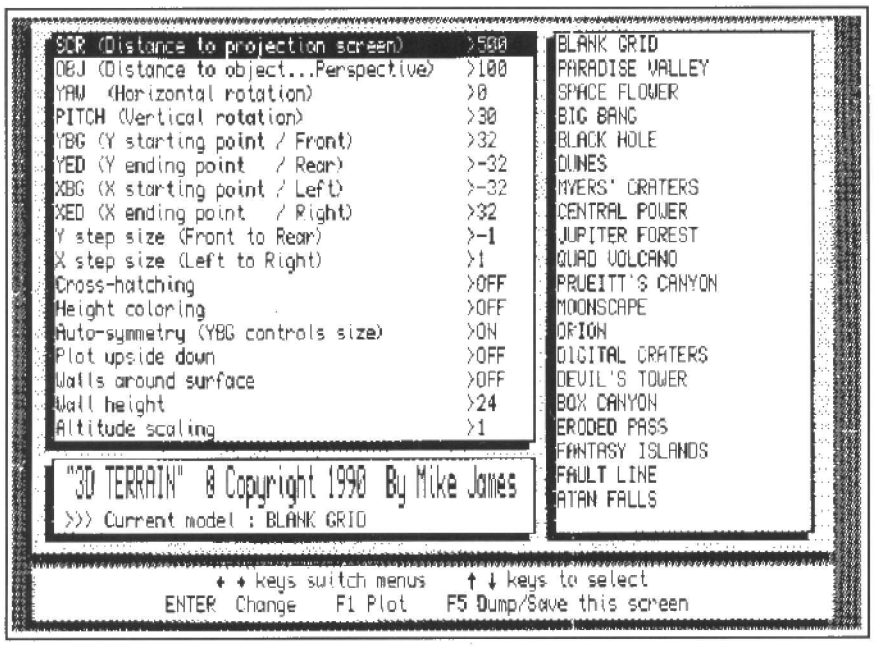
Clone

A rather odd clone routine is provided. It starts up by asking if both the original and destination 'cartridge' are in flp_2. Given the obvious answer 'no', it stops. If you answer 'y', you are then asked if you want the cartridge in flp_2 formatted; if the answer is 'n', you are returned to the first question. Fortunately, the COPY and WCOPY command will produce a working program copy without fuss.

Instructions are supplied on two *Quill* _Doc files — about three pages in all. To a large extent, you can go ahead using only the information given on the main screen **(see figure one);** you won't know that effect changes in parameters will have until you have tried them, anyway. The basic on-screen instructions are quite clear, and sufficient to get going with. The instruction files give some idea of what went into the writing of the program, and make the uses of the various parameters fairly clear. The emphasis is on enjoying using the program; it is, perhaps, a program that typifies what most types of micro computer have been 'about' for the past few years.

Extensions

The program is Turbo-compiled and a set of run-time extensions are loaded by the boot; the LINK_LOAD command is used to load various program modules. The program is nominally multitasking but, in practice, it is not conveniently so (on a JM QL at least). To be able to switch in and out of it, you have to be on a screen with a cursor, which the main menu screen does not have. If the program is started after something else has been loaded, the



Left: a sample screen from 3D Terrain. Far right: a meshed shape, ready to dump to the printer. Printouts are not as dramatic as the effect on-screen.

main menu may come up with the cursor in the normal SuperBasic position, from which one cannot switch into the program.

You can use Modes 4 and 8. The latter makes the drawing of pictures take longer, but it does give an added quality to them, justifying the suggestion of 3D in the program name. You can opt to have a black-on-white or white-on black drawing. 'Cross-hatching' (lines at roughly right-angles to the basic drawing lines) can be used, and it adds detail to give a somewhat less coarse-looking picture. 'Height-colouring' is an option which changes the drawing mode from mono to Mode 4 or Mode 8 colour; this tends to make pictures look more interesting, to the extent of seeming quite different in some cases. The pictures look much more realistic, and the shading more subtle, on a mono screen; the relative coarseness of colours gives bright pictures but tends to destroy the impression of real landscape.

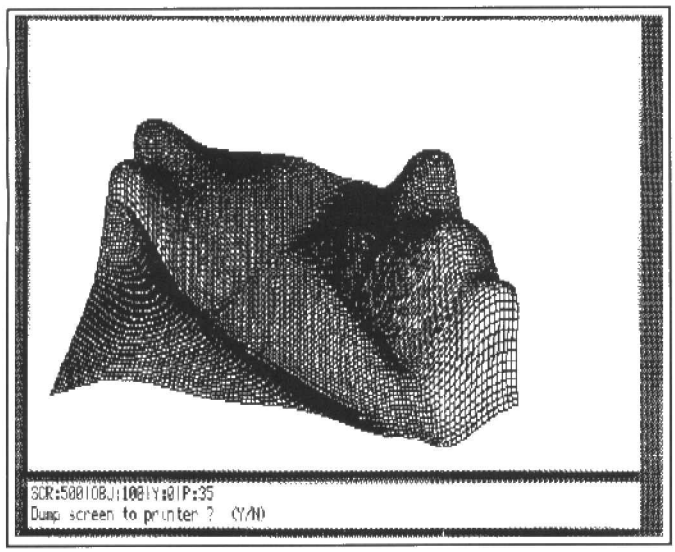
Samples

There are 19 'clip art' terrain pictures, and a blank grid pattern, supplied with the program. These vary considerably in their interest, and also in the time taken to display them. They are provided as bases upon which the user can develop his/her own pictures, by making use of the good selection of parameters. More 'shape libraries' may be made available by the program writer in future.

The first illustration shows the default setting. It is worth running through the sample pictures with these settings, to get an idea of what you might fancy modifying. Making large changes in certain parameters can result in apparent death of the program, with no lines being drawn on the screen over long periods; the message is, keep the changes small at first.

The first two parameters on the menu set the effective distance to an imaginary projection screen and the distance in front of it that the picture landscape is drawn. Varying these two changes your position, as the viewer, relative to the landscape. The next two parameters allow the entire landscape to be rotated horizontally or vertically; these will also alter the position of the picture within its window and you have to be careful not to force the bulk of it outside the window. The next four determine which part of a landscape is displayed, and the two following them set the effective fineness of the drawn pattern. The latter settings are a major influence on the time taken; the default setting of 1 for both X and Y step sizes give the characteristic chicken-wire appearance and allow reasonably speedy drawing, whereas reducing the step sizes to values less than 1 decreases the effective size of the mesh and correspondingly increases the drawing time.

'Cross-hatching' does what you would expect. Generally, having this set to 'on' improves picture quality, at the expense of increasing the drawing time somewhat. Height-colouring is a more complex function. When set to off, the Ink colour is black and the Paper white, and the Mode is 4. This avoids the mess printers can produce if the background is largely black. Setting this parameter to 'on' causes Paper to be black and Ink white, and the Mode can be either 4 or 8. The latter is lower resolution but, nevertheless, the additional colour gives an impression of increased detail. This may not appear to be logical, but it follows what has been noted with photography; the human eye can be convinced that a picture is 'sharp' even when there is a lack of detail, provided that a sufficient range and degree of colour is present.



Facilities

'Auto-symmetry' is linked to the starting/ending point settings. If it is turned off, the balance of lines drawn either side of the zero points can be altered. 'Plot upside down' is fairly obvious. 'Walls around surface' and 'wall height' control the drawing of a boundary around the landscape.

'Altitude scaling' permits the number of shapes appearing in a picture to be altered, while the relationship between individual shapes is maintained. The example quoted is of mountains being changed to molehills, and vice-versa.

Running the program on two QLs, side-by-side, it was possible to get a real feel for what the Gold Card does. Timings showed pictures being drawn at 3–4 times the speed on the GC-equipped system, compared to the system with an older (slower) *Trump Card*. Subjectively, what matters is that you can see lines being drawn quite briskly all the time on the faster system, whereas lines seem to track a snail's progress on the other. Nevertheless, the pictures get drawn on the TC system at a pace which is in line with what one expects on the QL with graphics programs. Some of the supplied pictures are simple enough to be completed on the screen in a few minutes on the TC system, but others take hours (depending largely upon parameter settings).

The built-in screen dump routine should give reasonable pictures on Epson-compatible printers. It certainly worked satisfactorily on a Kaga-Taxan DMP and Epson GQ-5000 laser printer (in FX80-emulation mode). The display screen size was made compatible with Digital Precision's *UltraPrint*, and it is stated that the 3D Terrain screens can be edited in almost any graphics program, as they are saved in standard Sbytes format.

Printing

Printed output is a fairly faithful rendering of what is shown on-screen. It is not quite as interesting as the 8-colour display, which possesses a certain 'luminance' in areas where there are sharp changes of direction; this can make both colour and mono screens look something out of the ordinary. The second illustration is actually a screen dump, rather than a print from the program, the reason for this being that the program prints are limited to FX80 resolution which gives a very flat, low-contrast image on paper when the screen picture is coloured. The dump (to a laser printer) is in LQ2500 resolution. When white-on-black pictures are printed, this printer mode gets rid of unwanted white lines and gives greater contrast but it is too dark overall. As always, printing is the frustrating area.

Instructive

To some extent, using this program is akin to playing a game, but it is also instructive. It might well be interesting to children who are being introduced to computing and mathematics or drawing. The ability to rotate objects and colour them, gives a useful degree of three-dimensionality – always welcome on our usually very flat screen. You get some of the feeling of expensive 3D drawing programs (but you can't draw directly yourself). The printed pictures are rather disappointing, when compared to what is seen on-screen, but this is a reflection on printers and the driver, not on the basic program. The program is easy to use and seems quite solid. It is slow and benefits considerably from the use of speed-enhancing devices such as the *Gold Card*, *Lightning* etc.

SOFTWARE FILE

Rich Mellor probes a **new** program using Ergon Development's Basic System Development Menu System.

MASTERBASIC

INFORMATION
Program: *MasterBasic* 1.31
Publisher: Ergon Development, c/o Davide Santachiara, via Emilio De Marchn. 2,42100 Reggio Emilia, Italy.
Price: £25 (£27 on mdv) plus £5 p&p.

MasterBasic (MBS) is the latest program to be released by this prolific Italian software house. As with the other programs from Ergon Development, it has been written in Turbocharged Basic and uses their Basic System Development Menu system. This ensuret the program is both quick and easy to use (see the review of QL Library Manager in QL World, August 1991 for further details).

This program is intended to provide a massive number of tools for the Basic programmer, from string search functions to a simple trace utility, and incorporates several features for which you would normally have to load the Basic program into an editor.

The program comes with a massive A4 manual (recently re-written), which goes in some depth into the variety of features. This new manual is well written and a massive improvement over the earlier English version, which suffered from having been translated from the Italian. Unfortunately, the manual could do with an index, which I believe may be produced once the program's development has settled down a little.

MBS loads quickly, and you are first presented with a title screen which details your name and address. When you press the space bar, a small warning window is displayed while MBS initiates various tables based on the Basic program already in memory. It then sits in memory until you need it.

The utilities provided by MBS are accessed by pressing the function keys (F1 to F5) either on their own or in combination with Shift or Ctrl. Although there is no provision to redefine these keys, it is relatively easy to put MBS 'to sleep'. This is achieved by calling the main F1 menu and selecting the Quit MasterBasic option (either using the cursor keys plus space, or by pressing the letter F This then calls up a small window in which appears the word Sleep. Altering the first letter to Q will remove MBS altogether, otherwise it is put into its sleep mode. In sleep mode, MBS will only recognise the key combination Ctrl Alt F1 which re-awakens it, leaving all the other function keys free to be used in other programs (*Quill* for example).

Tables

Ctrl Alt F1 also serves another function, in that if MBS is awake when these keys are pressed, it forces MBS to re-initiate its tables (if this is possible). This key combination must be used if MBS is to recognise any major changes to the current Basic program loaded in memory (such as another program having been loaded!); however, MBS will not allow you to use this if any of its menus are active or are 'suspended' (menus can be suspended by pressing Ctrl F1 so that you can access Basic and without losing the current menu).

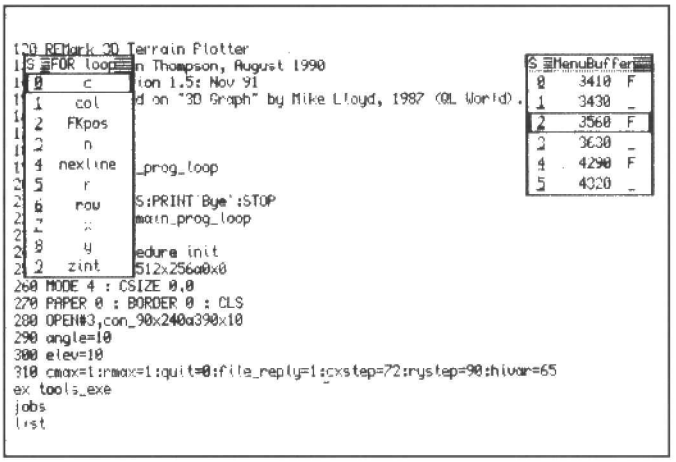


Figure one: Editing FOR loops using MBS

Tailor

During initialisation, MBS will tailor itself to your system. This enables MBS to decide whether it should use *Toolkit II*'s ED command, or have to resort to the slightly less useful Edit command; and whether it needs to save its own windows or whether it can rely upon Qjump's Pointer environment. MBS also copes well with *Minerva* and allows the user to switch integer tokenisation on or off, although due to the limitations of the current *Turbo* compiler, it cannot work with Minerva in dual screen mode. With regard to integer tokenisation, MBS did show up one slight problem with programs which have been saved using Liberation Software's *Qsave* utility – if you Qload a program with integer tokenisation in a different mode, then you can end up with numbers in a program in both integer and floating point form even though they look the same on screen (this is avoided if the Basic LOAD command is used, since this reformats the tokens to the current status). People who own Minerva and Qsave will therefore need to be more careful with integer tokens than other users. Unfortunately for users of Minerva (post 1.79), current versions of MBS cannot work on a MultiBasic program, since it only recognises the program loaded into the main interpreter (Job 0).

Once MBS has been initiated, the utilities provided by the package fall into four main categories: programming utilities, accessories, debugging tools and reference tools.

The accessories are called up by pressing F2. A menu is then shown which allows you to call up a simple calculator (which can work in decimal, hexadecimal or binary); a small notepad (the contents of which can be saved to a file for later use and utilities to enable you to find out the Ascii code for a given character or the keyrow value of keys.

The debugging tools are available on F4. These allow you to view the return stack (which contains information about the call to the last Procedure or Function); have details of the current line being executed and memory use while a program is running; and a profile op-

tion. The profile option is most useful to people who intend to compile their program, since it shows which parts of the program are executed most often. This feature can be restricted to run in certain Procedures/Functions, or you can specify a line number range over which it is to run. If you chose the former, then MBS quite helpfully presents you with a menu which lists all of the names of the Procedures and Functions used by the current program from which you can select the names of the Procedures/Functions to be used. Even more helpfully, this menu (like other Procedures/Functions menus) allows you to view a listing of a Procedure by using the cursor keys to move the cursor bar onto the desired name and pressing TAB.

Tools

The reference tools are the most powerful option of this whole package (and are certainly more powerful than any other similar program I have seen). If you want to find where a Procedure/Function is actually defined there are several ways in which you can do this. The easiest way is to press Shift F1 – there is then a short delay while MBS checks for the names of all Procedures and Functions defined in the program, and then these are listed alphabetically in a menu. Move the cursor bar onto the desired name (this can be done by using the cursor keys or by typing the name letter by letter) and MBS will then call up the start line of the definition by issuing an EDIT command (or ED if Toolkit II is present). If the definition line (eg DEFine PROCedure Junk) has been deleted for some reason, then a small warning window is displayed. Shift F2 allows you to list certain chosen procedures (for example if you wanted to check on how it works without editing it), but you can also 'reference' Procedures and Functions by using the F1 main menu and selecting Reference. This then goes through a similar process to allow you to select the Procedure/Function to reference, and then a small line menu is displayed which lists the definition of the Procedure/Function followed by a list of all the line numbers where the program refers to that Procedure/Function.

This reference sub-menu also allows you to find references to the different types of variables used by a Basic program (floating point, strings, integers, arrays, etc.) as well as machine code Procedures and Functions. As with referencing Procedures and Functions, once you have chosen the variable type, a small menu is shown on screen containing the names of all the variables of that type used in the current program, allowing you to pick and choose as you wish. If you do not like this method, then you can also just enter the name of whatever it is you are looking for and MBS will find all references to that name (be it a variable or machine code Procedure) in the program. Again a line menu is then presented with a list of all line numbers which contain references to the chosen variable.

The line menu is quite a powerful little extra to MBS's options in that not only does it contain all references to the given name/variable/Procedure, etc., but if you chose a line number using the cursor bar, once you have edited that line you can either press F1 to return to the line menu, or press Ctrl up or Ctrl down to go to the line number above or below (respectively) the current one in the line menu. This makes it very easy to quickly edit all occurrences of one variable for instance.

MBS is made even more powerful by the ability to restrict the search and reference utilities to certain Procedures/Functions in the program. However, other utilities allow the programmer to find any string within a program (a string needs to appear in the program between quote marks or after a REMark statement), something which is missed by all other utility programs on the market. Other options provided by the program allow you to find any given number or keyword (eg GO TO); again something else which is sorely lacking from other packages. This is further enhanced by the ability to link several of these things together so that MBS can be used to find something as complex as all references to 'IF sound-off THEN GO TO 1000'.

Token

A recent addition to MBS has been an ability to search for all FOR or REPeat loops in a program. Although there is a token for FOR/REPeat loops, because of the design of the SuperBasic interpreter, looking for this token is generally a waste of time since the token is only ever set after the program has been run and the given FOR/REPeat loop has been executed. Instead, if you press Ctrl Shift F3 (FOR loops) or Ctrl F3 (REPeat loops), MBS spends a little time examining the source program in memory to find all of the FOR or REPeat loops which are used by it. The time spent doing this is quite short even for a lengthy program, but it is certainly well worth it. MBS then displays the names of all of the REPeat/FOR loops in a menu and invites you to select one. Once you have chosen a name, a line menu is then produced which lists all of the lines where this loop appears. To assist you in making sense of the program (eg a program which uses lots of FOR i=1 to 100), the line menu contains symbols next to each line number to specify whether the loop name was preceded by REPeat, END REPeat, NEXT or EXIT. This option is extremely powerful and very useful, it is however a relatively new development and the way in which it is implemented could do with a little tidying up.

There is a separate Buffer Menu which can be used by the user and is similar in operation to a line menu. The user can however specify which line numbers are to be inserted in this menu and even place a three letter comment against each line number to remind him/her of why it is there. This can be useful for editing several parts of a program which are inter-dependant. Other options to help the programmer are the ability to place a whole program line into the MBS buffer (the contents of this buffer are then typed into #0 when Ctrl F4 is pressed), and thus merge two program lines together easily. You can even request MBS to place in the MBS buffer details of the definition line for a Procedure/Function – this is useful as a reminder of what parameters need to be passed to the given Procedure.

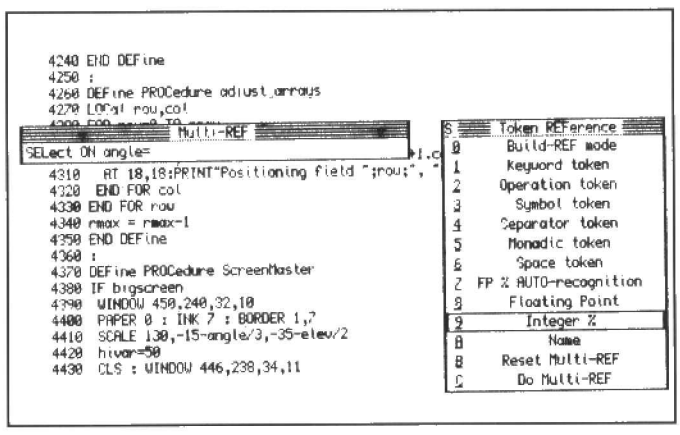


Figure two: Looking for a sequence of commands

Talk to us

There are many more powerful utilities provided by MBS, although many of them are more suited to the more advanced Basic programmer. However, the package is extremely easy to use and well thought out, and even the more casual Basic programmer should get a lot of use out of it. There is very little more that a programmer could need to assist in the development of programs and I am certain that if anyone has any more ideas, Ergon would be only too eager to hear them.

Finally, when sending for the package, please note that cheques should be made out to D. Santachiara and a further £4 must be added to the cost if sending a Eurocheque in any currency other than Lire.

Of Silver and Gold.

INFORMATION

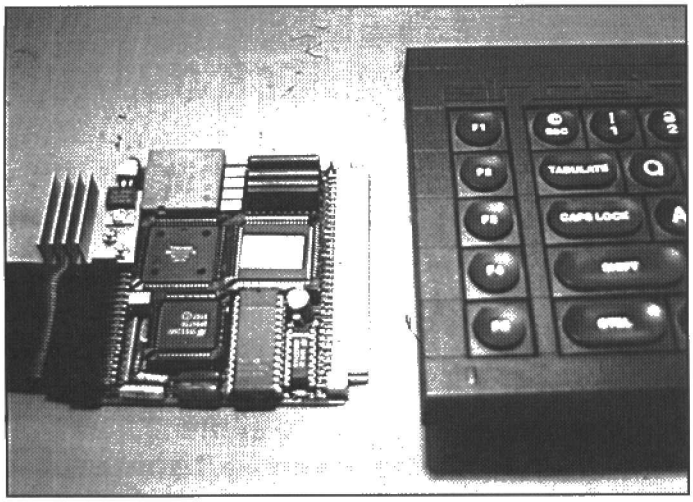
Product: Gold Card memory expansion card for the QL

Supplier: Miracle Systems Ltd, 5 Broughton Way, Osbaldwick, York. YO1 3BG, UK

Tel/fax 0904 423986

Price: New low price! £225 (export £200). Part-exchanges no longer accepted.

For further information about the 3.2 megabyte dual disk drive, see *QL Scene* and Miracle Systems' own advertisement.



The wait has only been due to the Gold Card's popularity - here at last is Mike Lloyd's assessment.

The Sinclair QL was given generous praise for being based on the Motorola 68000 central processing unit (CPU), even though it used the least powerful member of the family, the M68008 with its narrow eight bit data bus. It was believed that new QL models could take advantage of developments to the Motorola range as they became available. However, Amstrad's purchase of the Sinclair stable and their steadfast refusal to invest in the QL put paid to those hopes. The Motorola 68000 series now boasts a 68020, a 68030 and a 68040, none of which have seen the surface of a QL's motherboard.

Now, Miracle Systems have developed an ingenious and extremely powerful hardware extension which amounts to a heart-and-lungs transplant for the QL. Modestly marketed as the Gold Card, as though it as merely a development of their impressively successful Trump Card hardware, the new product brings the QL into the 1990's with a bang and, incidentally, overtakes the vast majority of budget price pcs in today's marketplace.

The new heart is a full 16-bit Motorola 68000 running at no less than 16Hz, compared with the QL's normal operating speed of 7.5MHz. The new lungs expand the QL's total memory capacity by a maximum of 768K. Various ancillary improvements include Tony Tebby's Super Toolkit II, a battery-backed clock to act as a permanent pacemaker and a disk controller which supports the brand-new 3.2MB extra-high density diskette forma. The whole package has been miniaturised onto a 10cm by 12cm card which slots unobtrusively into the expansion port on the left-hand-side of the QL, leaving just a small gold heatsink visible.

As a computer surgeon of very limited expertise, and even less confidence, I approached the transplant operation with extreme caution and some trepidation. But I found that the operation considerably easier than expected - this being non-invasive surgery, there was no need even to unscrew the QL casing.

Acting as my own theatre sister, I began by artfully unpacking the new implant from Miracle's sturdy packaging. The outer box (which perhaps should have been marked "With Care - Vital Computer Organs Inside") contained a much smaller inner box suspended firmly between two substantial pieces of foam. No flurry of polystyrene snowflakes to contend with here, thank goodness.

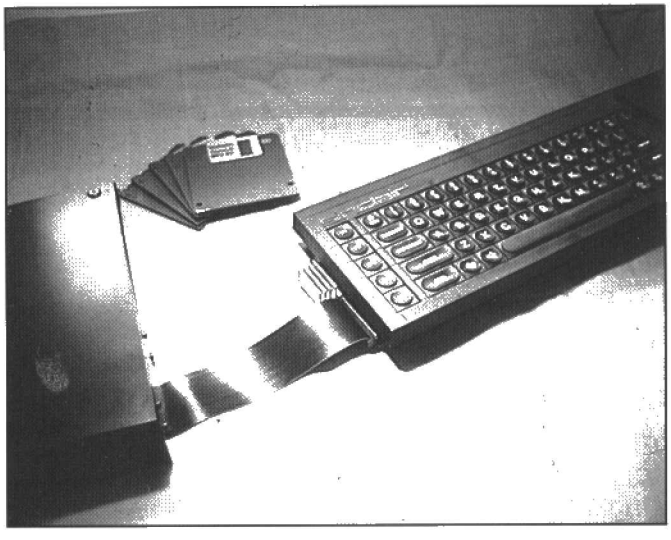
Nestling in polystyrene grips within the inner carton were a gold box marked "QL Gold Card", a double disk drive unit, a batch of five extra high density floppy disks and a comprehensive User Manual. Printouts of instructions for the card and for the disk drives ere folded neatly alongside their subject-matter.

The Gold Card is crammed with chips, although room has been found for a hefty three volt lithium battery to support the real-time clock, and for a chunky gold-coloured heat sink.

The operating theatre was prepared and the patient wheeled in. The QL already had early Expanderam and disk controller cards from Miracle Systems which were delicately extracted from the expansion port and set to one side. It was then a simple matter of pushing the Gold Card firmly into contact with the expansion port bus. This can sometimes be easier said than done, and care must be taken to ensure that the delicate pins are not bent out of true, but on this occasion everything went extremely smoothly.

Once in place, the Gold Card does not protrude at all from the QL casing. The parallel cable to the disk dries as easy to plug in, especially as a plastic lug ensures that it is impossible to fit it the wrong ay. The operation was over within seconds.

Miracle recommend reviving the patient with a single, bold burst of power applied from the mains socket via a



distribution board to all areas of the computer system. Alternatively, those of a more nervous disposition may prefer to power up the peripherals first and turn on the QL last. Miracle warn against the bad practice of removing the power cable from the back of the QL to turn the machine off - always keep the power cable attached to the QL and turn off at the mains.

The boot sequence provides an instant display of *deja vu*. The random dots which briefly appear turn the memory check suggest that only the first 128K is checked before the Gold Card interrupts matters and copies the QL's rom into its on, much faster, memory. It matters not the version or source of the rom: Minervas are just as acceptable to Gold Card as are the original Sinclair range of rom. The new ram-based operating system then begins its on boot sequence with its on memory check. Thanks to a high-speed algorithm, Gold Card can check its maximum of 1920K of ram in a couple of seconds.

Once booted, Gold Card exhibits the behaviour of a standard QL, but considerably accelerated. Using the 3.2MB extra-high Density (ED) disk drives supplied by Miracle, Psion's will could be loaded in four seconds. Even Digital Precision's enormous Professional Publisher with all its attendant resident extensions, high definition fonts and Lightning utilities can be up and running in about three-quarters of a minute.

Professional Publisher's nimble performance is a superb advertisement for the advantages of Gold Card. Pro Pub has always been an extremely powerful program, but the QL has seemed to be an inadequate vehicle for it. Screen scrolling as slow and jerky, some of the menu selections involved an irritating pause and negotiating through the mode options could be downright frustrating. Gold Card has put paid to all of that with a vengeance. The screen scrolls rapidly and smoothly, the sidebars are redrawn almost instantaneously and moving between modes is as fast as could be desired.

In fact, Professional Publisher running with a Gold Card in support is now, to my mind, superior to, say, *Draw Perfect* running on a full-blown 386 PC. Its true challenger in the pc market, however, its the renowned *Aldus Pagemaker*, which again would need a powerful computer to reproduce the responsiveness provided by the QL.

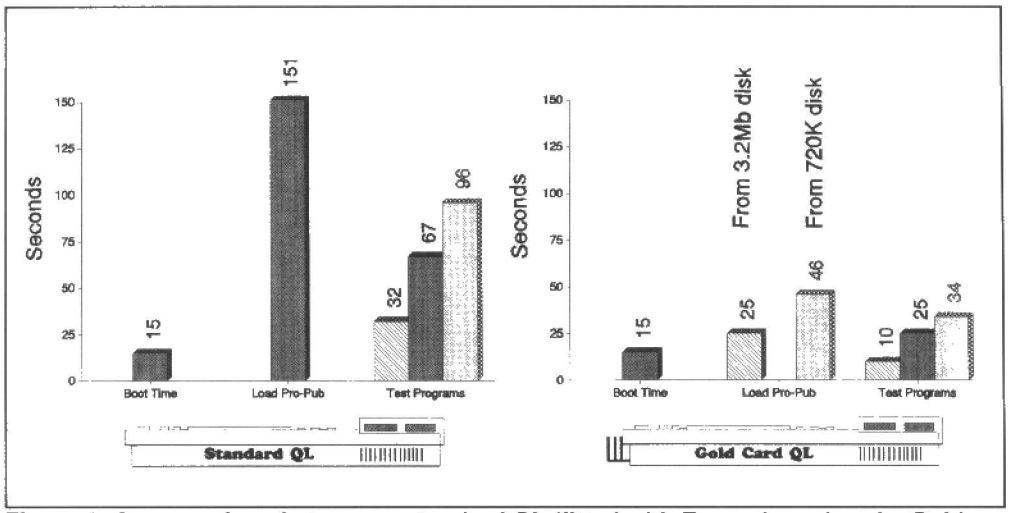


Figure 1: A comparison between a standard QL (fitted with Expanderam) and a Gold Card QL (fitted with 3.2Mb disk drive). The test programs were representative SuperBasic routines.

A further delight is the ability to load more high-definition fonts into the Gold Card's hue memory and still have plenty of room for some very large page formats which can be printed on A3 sheets and shrunk down to A4 size or below for printing. This technique creates high quality output quite acceptable for publishing.

Intrigued by Gold Card's ability to supercharge lethargic software, it seemed a good idea to road test perhaps DP's most complex product, *PC Conqueror*. As one of Conqueror's first reviewers, I had been immediately impressed by the technical aspects of this PC emulator, but less sanguine about its practicality. There are credible reports that when Conqueror is running on an Atari QL emulator it is faster than a native Atari PC emulator. Sadly, this is difficult to believe when Conqueror runs so ponderously on a standard QL.

With Gold Card added to a QL system, the emulator is running within seconds, with DOS taking slightly longer to boot up. To provide some hard statistics I ran the Checkit program disowned to test and benchmark PC hardware. The program was successfully fooled into thinking that it was running on an 8088 PC, with a BIOS of unknown origin. Somewhat bizarrely, Checkit reported that the CPU as running at 1102.31MHz but with a more reliable benchmark index of 101 Dhrystones. A standard IBM XT runs at 300 Dhrystones, whilst the more advanced IBM PS/2 Model 30 canters along at 800 Dhrystones. The screen refresh rate, however, showed a larger performance gap. The QL notched up a niggardly 202 characters per second whilst a nearby Elonex 386 managed 16,128 characters per second.

When MS-DOS programs were run under Conqueror with Gold Card in support they appeared to be quite usable. Games such as Blockit ere perhaps too slow to provide much of a challenge, but practical software such as the Galaxy Lite word processor ran at a perfectly acceptable speed. QL owners who have to work in the MS-DOS format to stay in line with their office computers can confidently bring tasks home to finish on the QL.

Gold Card comes with a complete version of Super Toolkit II embedded into an eprom chip. Most serious QL owners will already own this essential set of SuperBasic and qdos extensions, but having it on Gold Card means that the eprom port at the back of the QL is free for other software. All of the extensions developed for Miracle's hard disk are also included. A clutch of extra keywords specific to Gold Card compete the picture, improving the QL's ability to handle directories and coping with software which dislikes high speed and high memory capacity.

The MAKE_DIR keyword is used to create a subdirectory on a floppy disk or microdrive. It exists as both a command and a function, the latter for those programmers who need to trap for any file-related errors. MAKE_DIR uses filename prefixes to roup together files so that they ma be treated as though they ere the only files on a drive. Subdirectories created in this ay can be associated with a logical device driver called DEV to distinguish it from FLP, MDV and RAM.

Having created subdirectories, up to eight of them can be assigned to drives dev1_ to dev8_ simultaneously using a form of the DEV_ USE command. DEV_USE 3, flip_data_ will, for instance, allow you to list all files hose name begins with flpl data using the command dev3.

This facility is much more useful when Psion software is used with subdirectories. Unfortunately, Super Toolkit II subdirectories cannot be used with the Psion quartet de to their developers' fixation with MS-DOS and its ridiculous eight-character filename limit. By declaring a subdirectory with MAKE_ DIR and then assigning it a drive letter the Psion programs are confused into thinking that the QL no has a new type of drive and happily saves files to them

and loads files from them. With the command LU mdv this compatibility can be extended to software which only recognises microdrives, for instance allowing the SuperBasic INSTALL BAS program to load from a subdirectory and install new printer drivers into other subdirectories.

The new device driver also mimics the MS-DOS concept of a directory path. A path is a sequence of directories which are searched until a particular filename is discovered. For instance, EXEC_W quill will fail if the Quill executable file is not in the current directory defined by PROG USE. By declaring a series of subdirectory devices and appending an integer to them they form a search path, such as:

```
100 DEV_USE 1, flp2_, 2
110 DEV_USE 2, flp1_psion_, 3
120 DEV_USE 3, flp1_propub_, 1
```

The EXEC_W quill command ill no cause the current PROG_ USE directory o be searched, followed by flpl_ propub , flp2_ and flp1_ psion_, in that order. As soon as a file called "quill" is found it will be executed. DEV_ USE declarations are not permanent and so need to be included in a boot file or executed explicitly prior to them being needed. They can be revoked using DEV_ USE without parameters.

The opportunities for self-confusion are lain with PROG_ USE, MAKE DIR DEV_ USE and FLP_ USE all competing for the honour of providing access to files. However, by concentrating on DBV_ USE and either ignoring the other commands or tying them in with DEV devices almost all the potential problems can be avoided.

Gold Card sets sensible defaults for the disk drive step rate, but these way need to be altered to accord with the equipment you possess. The FLP_ STEP command sets the step rate in milliseconds, the defaults being 3ms for 80-track drives and 6ms for older 40-track drives. Different devices can have different step settings. For most users, however, the defaults ill be perfectly adequate.

Some programs, notably *Psion Chess*, refuse to operate with more memory than the 128 expected from a standard QL. Gold Card can fool software into thinking the QL has any amount of rams less than its actual complement by issuing the RBS SIZE command followed by the number of kilobytes required. Interestingly, RES SIZE prevents you from using extra-high density diskettes, presumably because there would be insufficient room for the Qdos file location tables. It does allow you to play Chess, however.

Some programs, notably action games, can object to the high speed of the Gold Card's C, or they might just become unplayable because everything happens so fast The wonderfully-named SLUG command imposes compulsory wait states so that you can adjust the processing speed to suit the software in hand.

Gold Card's battery-backed permanent clock is a godsend for date-relate (programs. The effort involved in setting the time and date on a normal) QL is usually out of all proportion to the use made of it, therefore most people simply do not bother. No, the date can be set once and it will remain reasonably accurate for up to five years. The date setting on the Gold Card can be protected so that changes to the SuperBasic date have only a temporary effect.

The Gold Card sent for review as accompanied by a brand new 3.2MB disk drive from Miracle. This new floppy disk format is presently unavailable for MS-DOS computers, although it should be operating on a number of Unix systems by no. Because QL software tends to be very compact, 3.2MB is more than enough for even the most sprawling suite of programs and their attendant data.

This makes the new disk format a very acceptable substitute for a hard disk. If, for example, all the Psion programs ere placed in their own subdirectories on a single extra density disk there would still be more than 2.5MB free for data directories. Access times are comparable with the Miracle hard disk. partly due to the compact data format on the 3.5in diskettes and partly due to Gold Card's improved CPU speed. The disk drives accept double density and high density disks in addition to the 3.2MB version, automatically detecting which format has been inserted.

The extra density floppy disks score over a hard disk drive in four key areas. Firstly backing up is extremely easy: WCOPY flp1 TO flp2_ will do the trick. Secondly, should disaster strike on a hard disk the potential loss could be devastating: with a 3.2MB drive only the current disk is at risk. Thirdly, whereas a hard disk is limited to, say, 32MB capacity, there is no limit to the number of 3.2MB diskettes in your disk library. Finally, the floppy diskette drives do not need a noisy and intrusive fan to cool their electronics.

Living with the Gold Card and the 3.2MB disk drives was extremely easy to get used to: high speed software loading, exemplary execution rates and enormous storage capacity on tap from the extra-high density disk drives add up to a computer user's heaven. In fact, it took a session with Gold Card removed to remind myself of ordinary QL performance before I fully appreciated how indispensable the Gold Card really is. The opportunity to reorganise my software library to take advantage of the extra density diskettes as a further blessing.

The final cherry on the cake as the resurrection of the old QL versus PC debate among my colleagues, this time with the QL very much in the driving seat. There are three desirables towards which the current generation of mainstream pcs are moving: filenames uninhibited by the eight-character limit imposed by MS-DOS, full multi-tasking, and a higher diskette capacity than the present 1.44MB. The QL has all of these features available no, and at a price considerably cheaper than can be obtained in the MS-DOS world. PC users are going to have to wait many months, and perhaps years, to catch up.

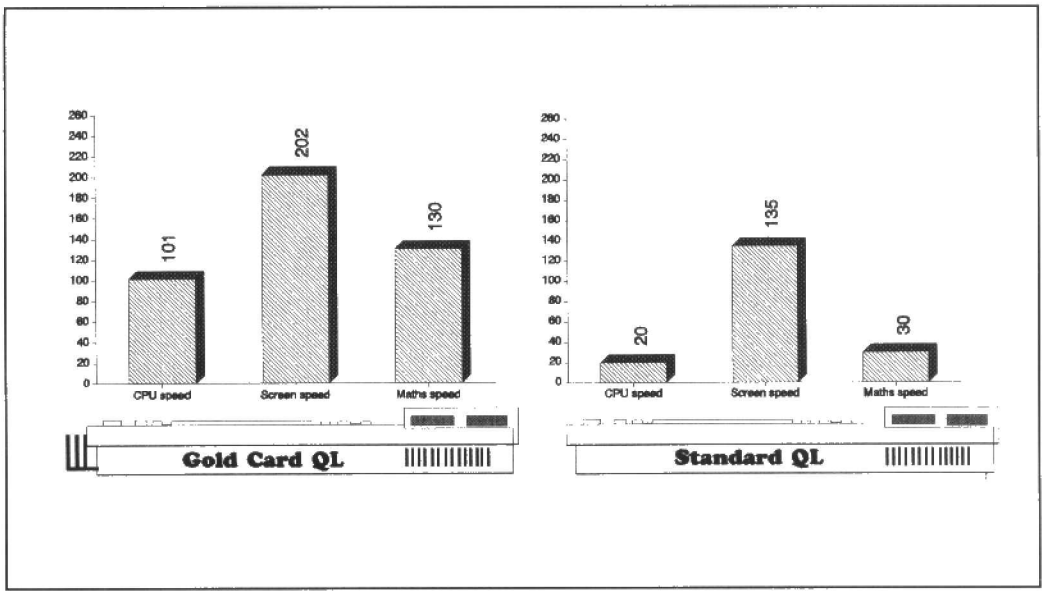


Figure 2: A comparison between a standard QL and a Gold Card QL running DR-DOS 5.0 under Conqueror. The test come from the Checkit benchmarking utility.

Archive Answers

Robin Stevenson puts tables on the list.

It is a sad fact that the most powerful of Archive's printing commands is also one of the least used or understood. Its power lies in its ability to automatically list the records in a file: printing them to screen, file or printer; show all or selected fields, and automatically space out the columns to accommodate the widest entry for each field. Its lack of use must stem in part from being blessed with the name DUMP.

Having a name which sounds like an unformatted hex print-out is a little unfortunate for a high level, highly formatted print command. Impressive though DUMP is, there are other reasons for its lack of popularity. The main one, I suspect, is the lack of control over the output width. Automatic spacing can be convenient, but if you have just one or two very long entries the lines will become impossibly wide, with acres of unused space. Another drawback is the need to spell out (and spell correctly) each field to include, unless you want the default of the whole lot.

Two new commands

This *Archive Answer* provides two new commands to complement DUMP in the task of getting a tidy print-out of your data. The design specification is to list all or selected fields, specifying the width of each column of data by simple on screen means; send output to screen, file or printer; print, at the top of each page of data, a heading of upper-case field names (without dollar signs), and cope with any file you care to throw at it.

The most obvious problem is what to do with text wider than the specified column width. One solution is to simply chop off any such text. The alternative approach is to allow text to spread over several lines, within the confines of its own column, as shown in **figure one.** Both solutions are offered here. The first, which I have called LIST, packs the most data in the least space, but at the risk of losing some of it. The second I have called TABLE. This offers a neater, more comprehensive solution, but in a rather less compact form. The names are slightly arbitrary in that LIST produces data in an undoubted tabular form, while TABLE's output is just as arguably a list. Hopefully the distinction is sufficient to allow the appropriate option to be used as required.

Listing one contains the LIST program. Almost all of these procedures will be used by TABLE as well. Only the additional procedures needed for TABLE are in **listing two.** The split in the listings is simply to allow you to enter it in two stages. The end result will be one program, providing the two commands. When entering the procedures, you should not type in the ':rem' sections which follow some lines. These are remarks, to help you understand what is being done. If entered they consume memory and drive space, and

```
proc CopyStruc;Oldfile$,Newfile$,Log$
  local COUNT,MAX
  let TEMPFILE$="TEMP_EXP"        :rem Use RAM drive if available
  use Oldfile$: let MAX=numfld()
  print : print "Copying structure to "+Newfile$;
  spoolon TEMPFILE$ export    :rem WRITE PROCEDURE TO CREATE FILE
  lprint "proc Temp"
  lprint "create '"+Newfile$+"' logical '"+Log$+"'"
  let COUNT=0: while COUNT<MAX
    lprint fieldn(COUNT)       :rem WRITE EACH FIELD NAME IN TURN
    let COUNT=COUNT+1: endwhile
  lprint "endcreate"
  spooloff : merge TEMPFILE$:Temp   ' :rem MERGE AND RUN THE PROC
  use Oldfile$: print
  endproc
```

```
proc Popen;File$,Log$          :rem ERROR TRAPPED VERSION OF OPEN
  open File$ logical Log$
  endproc
```

```
proc MakeWidth;Fname$,Log$           :rem GENERATE _WID DATA FILE
  error Popen;Fname$+"_wid","widths"     :rem ATTEMPT TO OPEN IT
  if errnum()=100:CopyStruc;Log$,Fname$+"_wid","widths"
    use "widths": append : endif      :rem IF NOT, CREATE NEW ONE
  use "widths": display : sprint
  print at 19,27;"Enter the column width for each field."
  print at 20,27;"For fields not required enter 0."
  print at 21,27;"For Numeric fields, use format width.decimals"
  alter
  endproc
```

```
proc Heading;Bold$      :rem GENERATE THE HEADING AND EXTRA LINES
  local COUNT,WID,WIDTH,T$
  let HEADING$=Bold$: let WIDTH=0
  let COUNT=0: while COUNT<numfld("widths")
    if fieldt(COUNT): let WID=val(fieldv(COUNT,"widths"))
      else : let WID=fieldv(COUNT,"widths"): endif
    if WID>0           :rem ONLY INCLUDE THOSE WITH A WIDTH VALUE
      let WIDTH=WIDTH+WID        :rem KEEP A RUNNING WIDTH TOTAL
      let T$=fieldn(COUNT)( to len(fieldn(COUNT))-fieldt(COUNT))
      if len(T$)>WID-1: let T$=T$( to WID-1): endif
      if fieldt(COUNT)=0   :rem NUMERIC HEADINGS JUSTIFIED RIGHT
        let T$=rept(" ",WID-len(T$)-1)+T$: endif
      let HEADING$=HEADING$+T$+rept(" ",WID-len(T$))
      endif
    let COUNT=COUNT+1: endwhile
  let HEADING$=upper(HEADING$)+Bold$
  if ExtraLine$<>""       :rem MAKE EXTRALINE$ THE REQUIRED WIDTH
    let ExtraLine$=rept(ExtraLine$,WIDTH): endif
  endproc
```

```
proc NewPage;FollowOn         :rem PAUSE/FORM-FEED + NEXT HEADING
  if FollowOn<2: if OUTPUT=1
      print "Press any key to continue";
      let ANY$=getkey(): print chr(13);chr(27)+"A";
      else : lprint chr(12): endif : endif
  if FollowOn: let LINE=0: lprint tab MARGIN;HEADING$
    if len(ExtraLine$): lprint tab MARGIN;ExtraLine$
      let line=
  endproc
```

```
proc AddNumber;Number,Wid
  if FirstLine          :rem PRINT BLANKS FOR OTHER LINES OF TABLE
    local DP: let DP=int((Wid-int(Wid))*10.1)
```

```
    if DP=0: let ANSWER$=ANSWER$+num(Number,Wid-1)+" "
      else : let ANSWER$=ANSWER$+dec(Number,DP,Wid-1)+" ": endif
    else : let ANSWER$=ANSWER$+rept(" ",Wid): endif
  endproc
```

```
proc PrintLine;Log$              :rem PRINT SINGLE LINE OF A LIST
  local COUNT,WID,DP,T$: let ANSWER$=""
  let COUNT=0: while COUNT<numfld("widths")
    if fieldt(COUNT): let WID=val(fieldv(COUNT,"widths"))
      else : let WID=fieldv(COUNT,"widths"): endif
    if WID>0              :rem ONLY USE THOSE WITH A WIDTH VALUE
      if fieldt(COUNT): let T$=fieldv(COUNT,Log$)
        if len(T$)>WID-1: let T$=T$( to WID-1): endif
        let ANSWER$=ANSWER$+T$+rept(" ",WID-len(T$))
        else :AddNumber;fieldv(COUNT,Log$),WID
        endif
      endif
    let COUNT=COUNT+1: endwhile
  lprint tab MARGIN;ANSWER$: let LINE=LINE+1
  if LINE=PAGELENGTH:NewPage;1: endif
  endproc
```

```
proc PrintList;Fname$,Log$,Opt$,Table  :rem USED BY LIST & TABLE
  if Log$="": let Log$="main": endif
  error Fopen;Fname$,Log$: use Log$     :rem ENSURE FILE IS OPEN
  mode 0,8: print chr(20)+chr(4)+chr(1)+chr(76)+chr(24);
  if Table:MakeMerge;Fname$,Log$     :rem EITHER MERGE _TAB FILE
    else :MakeWidth;Fname$,Log$: endif     :rem OR GET _WID FILE
  if instr(lower(Opt$),"screen")       :rem SET UP OUTPUT DEVICE
    Heading;chr(26): spoolon-screen              :rem FOR SCREEN
    cls : print chr(25)+chr(1);
    let MARGIN=0: let OUTPUT=1: let PAGELENGTH=21
    else : if instr(lower(Opt$),"file")     :rem FOR EXPORT FILE
      spoolon Fname$ export : let OUTPUT=2
      else : spooloff : let OUTPUT=0: endif
    Heading;chr(15): let MARGIN=10: let PAGELENGTH=61: endif
  let LINE=0: use Log$: first :NewPage;2
  while not eof(Log$)          :rem LOOP THROUGH EACH DATA RECORD
    let FirstLine=1: if Table:PrintTable
      else :PrintLine;Log$: endif
    next Log$: endwhile
  NewPage;0
  spooloff : close "widths": mode 1
  endproc
```

```
proc List;Fname$,Log$,Opt$
  let ExtraLine$=""
  PrintList;Fname$,Log$,Opt$,0
  endproc
```

```
proc AddText;Text$,Wid
  local POS,SEP$: let SEP$=" ,.-;:=+?"
  if len(Text$)<Wid              :rem IF IT WILL FIT ON THIS LINE
    let ANSWER$=ANSWER$+Text$+rept(" ",Wid-len(Text$))
    let REMAIN$="": return : endif
  let POS=WID-1        :rem IF IT IS TOO LONG TO FIT ON THIS LINE
  while POS>3 and Wid-POS<9 and not instr(SEP$,Text$(POS))
    let POS=POS-1: endwhile      :rem FIND A SUITABLE BREAK POINT
  if not instr(SEP$,Text$(POS)): let POS=WID-2
    let ANSWER$=ANSWER$+Text$( to POS)+"- ": else
    let ANSWER$=ANSWER$+Text$( to POS)+rept(" ",Wid-POS)
    endif             :rem THE UNPRINTED TEXT IS PUT INTO REMAIN$
  let REMAIN$=Text$(POS+1 to ): let MORE=1
  endproc
```

```
proc Fmerge;Fname$            :rem ERROR TRAPPED VERSION OF MERGE
  merge Fname$
  endproc
```

```
proc MakeMerge;Fname$,Log$          :rem GENERATE _TAB PROC FILE
  error Fmerge;Fname$+"_tab"     :rem TRY TO MERGE EXISTING FILE
  if not errnum(): error Fopen;Fname$+"_wid","widths"
    if errnum()<>100: return : endif : endif
  local COUNT,WID,T$,NUMERIC        :rem IF NOT, CREATE A NEW ONE
  MakeWidth;Fname$,Log$
  spoolon Fname$+"_tab" export
  lprint "proc Temp"
  let COUNT=0: while COUNT<numfld(Log$)
    let T$=fieldn(COUNT,Log$)
    let NUMERIC=fieldt(COUNT,Log$)=0
```

slow the program down, so you are better off referring to the magazine for them.

CopyStruc will be familiar to anyone who studied my 'Totals and Averages' (*QL World,* 1991), program. It is exactly the same as the one in that program, so typing time can be saved by extracting that. Load TOTALS, and then delete all the other procedures. The other LIST procedures can then be added to it, before saving as LISTS. You could add all the *Archive Answers* together, to make a large toolkit program. While this would have its advantages, it would make loading very slow. On balance I prefer lots of small programs, which can be loaded as required.

To use LIST you enter it as a command line instruction, followed by a number of text parameters. The first of these is the file name. It doesn't matter if the file is open or closed at the time. LIST will open it if necessary. The second parameter is the logical file name. If you have no files open, you can send a null string ("") and LIST will use the default name of "main". Finally you must specify the output device. Again there is a default, to the printer. Otherwise you can specify "screen" or "file". If you choose the file option, output will be to a file with the same name as the data file, but with an "_exp" extension, in a form suitable for importing into *Quill.*

If you wish to list GAZET_DBF, for example, then assuming it is on your default data drive you would type:

```
LIST; 'GAZET', 'GAZ', 'SCREEN'
```

You can of course specify a drive as part of the file name. List requires a file to hold the widths of each column, in addition to the data file. This is of identical structure and name to the data file, but with a "_wid" extension. The first time you run LIST for a particular file, it will automatically create the width file. Once created, LIST will be able to re-use it, making the process rather faster on subsequent calls.

A list of the field names is displayed on screen for you to enter the width values. Enter the column widths you require in both text and numeric fields. There is always a space at the end of each column, so usable width is one less than the width value entered. Numeric fields can be in two parts. The integer part denotes the total width, as for the text, but you can also specify the total number of decimal places by putting this after the decimal point. For example a value of 7.2 would give a format of 'xxx.xx'. Any field with a zero, or no number at all, will not be shown in the list. If there are more than 24 fields, you will not be able to enter column widths for those. The total width across the page will be the sum of these widths. Except when outputting to the screen there will also be a margin of 10 spaces.

Once listing one is entered and saved, you should try it out on some test data. As with all programs, check it out thoroughly

```
    let NUMERIC=fieldt(COUNT,Log$)=0
    if NUMERIC: let WID=fieldv(COUNT,"widths")
      else : let WID=val(fieldv(COUNT,"widths")): endif
    if WID                :rem FOR EACH VARIABLE WITH A WIDTH VALUE
      if NUMERIC: lprint "Addnumber;";T$;",";WID
        else      :rem WRITE THE ENTRY, EITHER FOR NUMBER OR TEXT
        lprint "AddText;";T$;",";WID;": let ";T$;"=REMAIN$"
        endif : endif
    let COUNT=COUNT+1: endwhile
  spooloff :Fmerge;Fname$+"_tab"      :rem AND MERGE THE NEW PROC
  endproc
```

```
proc PrintTable                  :rem PRINT MULTI-LINE TABLE ENTRY
  let MORE=1: while MORE: let MORE=0      :rem LOOP UNTIL NO MORE
    let ANSWER$="":TEMP: lprint tab MARGIN;ANSWER$
    let LINE=LINE+1: let FirstLine=0
    if LINE>=PAGELENGTH:NewPage;1: endif          :rem PAGE BREAK
    endwhile
  if ExtraLine$<>"" and LINE>1
    lprint tab MARGIN;ExtraLine$     :rem EXTRA LINE, IF REQUIRED
    if LINE+2>=PAGELENGTH:NewPage;1     :rem AND CHECK PAGE BREAK
      else : let LINE=LINE+1: endif : endif
  endproc
```

```
proc Table;Fname$,Log$,Opt$,Extra$
  let ExtraLine$=Extra$
  PrintList;Fname$,Log$,Opt$,1
  endproc
```

before subjecting real data to it, and always have a back up copy as well. Note that the data file will be left open after use, even if it was LIST that opened it. This makes it even quicker for a re-run, perhaps to a different output, or with a different format. But you must remember to close it after you have finished.

TABLE is used in a similar way to LIST. All the parameters described for LIST are needed for TABLE, plus one more. Because TABLE entries can occupy several lines, it may become less clear which lines belong to which entry. There is therefore an option to insert a line between each entry. You need to supply the character to be repeated across the line – it could be a dot, dash, space, underscore or whatever. If you send a null string ("") the extra line will not be inserted.

Also like LIST, TABLE requires a _WID file for each data file, and can use the same one that LIST created. However in addition, TABLE needs another file, this time with a "_TAB" suffix. This file is a short program file, which is merged, and then run once for each line of the table. Again, if it is not found, TABLE will automatically create it.

This technique of developing secondary files for use alongside a data file is extremely useful. Each data file will require a small family of related files, all with the same name, but different suffix. Thus if you are using GAZET_DBF, you will find on the same drive GAZET_WID, GAZET_TAB, and perhaps GAZET_EXP. Equally importantly, if any of these support files are absent they can be automatically generated, with just a small time penalty.

The _TAB file is a program which 'hard-wires' the width values into the program. If there is a _TAB file present you do not get an option to alter the widths, unlike in LIST. To alter the widths you must delete the _TAB file (e.g. KILL "GAZET_TAB") before running TABLE. You can then revise the widths, which will be used to write a new _TAB file.

Because of the _TAB file, which is only merged at run time, it may be rather unclear how TABLE works. I shall offer a brief description here for those interested in such things:

The primary procedure for both TABLE and LIST is proc PRINTLIST. This sets the scene, via support procedures, by ensuring the _WID and _TAB files are available, that output is redirected as specified, and the heading$ and Extraline$ variables suitably filled. This done, the program loops through each record calling either Printline (for the LIST) or PrintTable. PRINTLINE fills and prints the various ANSWER$, to give one line of text, based on the data and width files.

Second loop

PRINTTABLE, on the other hand contains a second loop, controlled by the global variable (MORE). Having set MORE to 0, it calls the procedure from the _TAB file, which is called TEMP. After running the program, you can examine proc Temp, and will see it contains a string of calls to either AddText, or AddNumber – one for each field in the table. AddNumber is the simpler. If it is the first call for this record, it adds the number to ANSWER$, as a string of the required width. Subsequent calls simply pad out the line with the required number of spaces.

AddText on the other hand has to cope with string lengths of 0 to 255 characters. It starts by comparing the Text length with the required width. If it will fit, it is added to ANSWER$, along with such space padding as required, and then returns. If it is too long a suitable break point must be found. If none of the specified separators are identified, the word is hyphenated. In either case, the text to be printed is added to ANSWER$, the remainder put in REMAIN$, and MORE is set to one . Back in proc Temp, the value in REMAIN$ is stored in the appropriate field of the data file, and the next field is dealt with in a similar fashion.

CODE	DESCRIPTION	SOURCE	PRICE
514	Essex Flange - 42mm diam.	AST, Groby	5.64
517	Hot Water Tank, 120 litre, with integrated immersion heater.	PlumRite, Burleys Way.	84.70
592	Bath Tap - Hot	AST, Groby	12.32
593	Bath Tab - Cold	AST, Groby	12.32
602	Radiator bleed screw (8mm thread)	Just Rads, High Street.	0.83

Once all the fields are so treated, control passes back to PrintTable, which prints out the accumulated line. If MORE has been set to 1 the loop will continue, the temp will again be processed, using the now truncated record. If some of the fields are still too long, the process will be repeated as required. If any of the fields has to be truncated, MORE will again be set, and the loop continued. Only when all have been completed will the MORE loop exit. PrintTable then prints the extra line if required, and control passes back to PrintList to get the next record.

Astute readers will have noticed that during this process all the text variables in the record have been truncated out of existence. They are all now empty. Fortunately this situation is not permanent, or all your precious text would disappear at this point. The changes can only be fixed by an update command, which is not present. When the next record is accessed, the contents of the previous are restored to their former state.

Various global variables are used in addition to ANSWER$ and REMAIN$. OUTPUT specifies which device is being used. MARGIN is set to 10 for printer and file, but 0 for screen output. PAGELENGTH similarly provides for 61 lines on a page, 21 on the screen. LINE keeps track of the current line, to identify the page breaks.

It is hoped that LIST and TABLE provide convenient and useful ways of displaying and printing your data. PC-Archive users may be interested to note that they will run almost unchanged on that platform. (You will need to replace underscores with full stops in the File Names.) Both commands provide compact, clear representations of your data, and TABLE in particular puts it in a form which would be very tricky to achieve by any other means.

If you have any topics that you would like Robin Stevenson to address in forcoming editions of Archive Answers, please write to him care of QL World.

Are you a QL CRIMINAL?

The Data Protection Laws protect citizens from false data, but have they become too trivial? asks David Drysdale.

To be classified as a 'criminal' for storing apparently innocent information in your QL's microdrives would seem to be nonsensical - and yet this could happen and you could be subject to a £2,000 fine!

The laws you would be breaking are, of course, those under the Data Protection Act of 1984 which protect individuals from untrue or inaccurate information about themselves being recorded in data storage systems. This type of error could have damaging results such as the refusal of credit or bank loans.

Before these aggrieved individuals could take steps to have this false information removed or corrected they would, of course, need to know who was holding it, and so the law requires all data users storing personal information to register this fact and to abide by a code of conduct. An individual has the right of access to this information whether or not the data user is registered.

The penalties for refusing to abide by these laws can be quite steep and, in extreme cases, could even result in the police obtaining a warrant to break in to a property to seize the offending magnetic media.

Obviously, a large amount of material produced by computers, such as payroll information, wordprocessed letters and small business accounts poses no threat to anybody and the Act recognises this by granting exemptions from the need to register. These exemptions are of a very limited nature, however, as I discovered, to my cost, when I telephoned the Data Registration office while researching this article.

As a freelance journalist I put it to them that I would have no need to register as I only used a computer for word-processing and simple accounts.

However, like most journalists, I have a contacts book in which I record the names, addresses and phone numbers of people who have supplied me with information and I was thinking of keying this into my QL for rapid reference. Surely that was permissible within the exemptions?

I was wrong. I would certainly have to register, at a fee of £75 renewable every three years, if I computerised that name and address list. It was perfectly in order to keep it as a written file without registering, because the Act only covers data but the moment it became stored on a disk or microdrive cartridge I would have to register - even though it was only for my personal use and would not be disclosed.

This interpretation raises important questions for the thousands of users of computerised notepads and information-storing calculators which carry names and addresses, particularly for business use, and who have not even considered that they may need to register with a Government agency.

Even a program such as a computerised personal diary would not be immune from the Act if it contained information about any individual still alive. As the registration requirements are strictly for electronically stored data it would be perfectly in order to key in a list of names, addresses and personal information items and then instruct the computer to sort them into alphabetical order, say, before printing them on paper.

Once the printing out had been accomplished, though, the information would need to be cleared out of memory immediately, and no permanent storage could be used. Even storing the information with the sole purpose of reprinting the list at a later date would be unlawful unless registration had taken place.

To his credit, the Data Protection Registrar has sought the help of the Plain English Campaign in producing some easily understood leaflets to explain what the Act is about and he has also issued an excellent one-page flow chart to enable data users to decide whether they need to register.

One of the flow chart boxes asks: "Does any of your computer-held information contain details relating to living individuals?". If the answer is "yes" it goes on to ask "Is this personal information of a sensitive nature?" and to a "no" response it would give the rejoinder "This is not important. Personal information can be as little as a name and address". The document also makes it clear that exemptions only cover the simplest of tasks.

Upon registration data users will have to answer five questions about each datafile in use. These are: What personal data is being held? What is the purpose for which it is used? What are the sources from which the information may be obtained? To whom might the information be disclosed? And, finally, details of any overseas countries to which the data could be transferred would have to be given.

After registration, data users would have to adhere rigidly to eight Data Protection Principles which cover such areas as the legality of the information, its relevance, accuracy and updating and the standards of security under which it is kept. Persons named in the database would also have the right to obtain a printout of information held about them upon payment of not more than £10.

This right of the individual to know what is being held in the datafile is a major part of the Act, and I had an opportunity to put it to the test recently.

An inspector from the local Electricity board had paid me a visit and plastered my meter and fuse boxes with a mass of vivid yellow stickers warning that the equipment must not be tampered with. At first I thought these stickers were being applied to all meters but it appeared I was the only one on the estate to have them. I then became quite concerned because being selected in this way could indicate that I was under suspicion of tampering. I wrote immediately demanding to see my computer records.

At first I was asked for a £10 fee to see them, but the board relented when they heard the story and sent them along free. They also sent a covering letter confirming that nothing incriminating was in the records and stated that the stickers had been applied by an 'over-enthusiastic' employee who had since left. Would the electricity authorities, I wondered, have responded so rapidly without the existence of the Data Protection laws?

Even so, I feel that the valuable work of the Data Protection Registry is in danger of being undermined by its insistence that piffling little private datafiles used by the self-employed, and files kept in electronic Filofax-type pocket computers are within its remit. The emphasis should surely be on the disclosure patterns of databases rather than on the mere fact of electronic storage.

People not wishing to register can, of course, simply resort to paper files again, which seems to be a backwards step. In any case, printed documents can always be scanned by Ocular Character Reading equipment and fed back into computers as easily as if they were on disk.

It has been said that bad laws make bad citizens, and although the Data Protection laws are not in the 'bad' category they are certain to alienate those who will resent being labelled 'criminal' for simply transferring the contents of their trusty old paper diaries onto a QL disk or microdrive.

For information and advice on the Data Protection Act contact:

Information Services Dept., Data Protection, Springfield House, Water Lane, Wilmslow, Cheshire SK9 5AX. Tel. 0625 535777.

DIY TOOLKIT

Simon Goodwin extends the QL with code to search arrays hundreds of times faster than SuperBasic, and a demonstration of rom text comparison routines.

```
* QL WORLD DIY TOOLKIT - ARRAY SEARCH ROUTINES
* Version 0.7, Copyright 1992 Simon N Goodwin.
*
initialise lea.l    define,a1
           movea.w  $110.w,a2       BP.INIT vector
           jmp      (a2)
*
* INARRAY% ( array [ ,start% ] ,value ) scans arrays FAST
*
inarray    cmpa.l   a3,a5           Any parameters?
           beq.s    bad_param
           cmpi.b   #3,0(a3,a6.l)   Initial array parameter?
           bne.s    bad_param
           moveq    #3,d5           Mask for known SB types
           and.b    1(a3,a6.l),d5   Pick up datatype
           beq.s    bad_param       Exclude substring type
*
* D5 = type; A3 = Name Table offset of the array to be searched
*
           movea.l  4(a3,a6.l),a1   A1 -> Array descriptor
           movea.l  40(a6),a4       Fetch BV.VVBAS offset
           adda.l   a4,a1           A1 is descriptor A6 offset
           adda.l   0(a1,a6.l),a4   A4 is offset of values
           move.w   4(a1,a6.l),d0   Get number of dimensions
           moveq    #1,d7           Initial index multiplier
           cmpi.b   #1,d5           Check type; 1 means String
           bne.s    get_index
*
* Treat two-dimensional character arrays as a special case
*
           subq.w   #2,d0           Faster than CMP if volatile
           bne.s    bad_param       Wrong number of dimensions
           move.w   6(a1,a6.l),d7   Maximum element number
           move.w   8(a1,a6.l),d5   Stride of each string, 4+
           bpl.s    next_param      Disallow negative lengths
           bra.s    bad_param
*
* Numeric array descriptor; 1..15 dimensions, 1..32768 elements?
*
get_index  move.w   6(a1,a6.l),d2   Fetch maximum index value
           addq.w   #1,d2           Convert to stride
           mulu     d2,d7           Accumulate
           addq.l   #4,a1           Try the next one
           subq.w   #1,d0           Count one less subscript
           bne.s    get_index
           subq.l   #1,d7           Count elements from zero
           cmp.l    #32767,d7       Limit result to 0-32767
           bhi.s    bad_param       Alas the array is too big
next_param addq.l   #8,a3
           cmpa.l   a3,a5           Any more parameters?
           beq.s    bad_param
           swap     d7              Save count for later
           move.w   d5,d7           Save type: 2=F, 3=%, 4+=$
*
* D7.W = type; D7.H = last element, 0-32767; A4 = initial offset
*
           moveq    #0,d5           Assumed start, element zero
           lea.l    8(a3),a2        Delimit one Name Table entry
           cmpa.l   a2,a5           Is there one parameter left?
           beq.s    pattern         If so, D5 is set correctly
*
* Fetch the optional integer 'first element to scan' in D5.W
*
           move.l   a5,d5           Save end of parameters
           movea.l  a2,a5           Delimit the first parameter
           movea.w  $112.w,a2       Read vector to get integers
           jsr      (a2)            Call CA.GTINT
           bne.s    bad_exit
           move.w   0(a1,a6.l),d0   Pick up the value
           bpl.s    good_param      It needs to be positive
```

The newest DIY Toolkit project builds on the array routines introduced in April. INARRAY% is a function which can search string, integer or floating- point arrays for any value at breathtaking speed.

Arrays are tables of values referred to by a single name. Individual values, or 'elements' of the array, are distinguished by subscripts written in brackets after the array name. For instance, DIM TABLE(9) reserves space for an array of ten elements, TABLE(0) to TABLE(9).

TABLE is termed a one dimensional array because each element is located by a single subscript. QL arrays may have any number of dimensions, as long as there is enough room in memory for all the elements. DIM BOARD%(7,7) reserves one integer element for each of the 64 squares on a chessboard, so BOARD%(0,0) can record the contents of the top left square, and BOARD%(7,7) the bottom right. BOARD% is a two dimensional array.

Arrays are useful because they allow a single Basic routine to process each element in turn, varying only the subscripts. There is no need to re-write the program for every value, as would be the case if each element had a distinct name.

The snag is that programs which use large arrays can become very slow, as several statements must be interpreted in the course of processing each element. This overhead is repeated for every value, so the time taken to scan a large array can be hundreds of times more than would be the case if it was all processed by one powerful toolkit routine.

INARRAY% was inspired by the function of the same name in *BetaBasic* for the Sinclair ZX Spectrum, and its counterpart in Andy Wright's latest package, *MasterBasic* for the Sam computer. The DIY version is more powerful because it uses special-purpose code for each Super Basic data-type, scanning numeric arrays at around 60,000 elements per second on a 640K QL, and faster still on a 68000-based system.

INARRAY% does nothing that is not possible in normal interpreted SuperBasic, but it is so much faster, even compared with compiled Basic, that it allows new types of programs to be written and run at an acceptable speed. INARRAY% also provides a framework for programmers who wish to re-write the inner loop of a slow program in machine code.

Tests on a basic 128K QL show that INARRAY% is more than 250 times faster than interpreted SuperBasic when searching numeric arrays, and about ten times faster than optimised Turbo-compiled SuperBasic!

My QL manages 38,500 integer comparisons per second in slow ram, rising to 63,000 in uncontended expansion memory; that's 412 times faster than Sinclair Basic. The code to scan all types of array requires just 348 bytes of memory, and may be shared between any number of tasks.

INARRAY% is so fast that it can search a numeric array of 32768 elements in less than a second, regardless of the speed of your QL. This compares with 231 seconds for an equivalent loop in Sinclair

```
*
bad_param   moveq    #-15,d0          Signal a BAD PARAMETER error
bad_exit    rts                       Return the error code in D0
*
good_param  addq.l   #2,$58(a6)       Unstack integer from BV.RIP
            movea.l  a5,a3            Step past this parameter
            movea.l  d5,a5            Restore end of list
            move.w   d0,d5            Update D5 to match parameter
            lea.l    8(a3),a2         Only one parameter left?
            cmpa.l   a2,a5
            bne.s    bad_param
*
* Pick up the pattern and look for it in the array
*
pattern     cmpi.w   #3,d7            Check type
            bmi.s    floats           Type 2 (float) gives -1
            bne.s    strings          Type 3 (integer) gives zero
*
* Search integer array elements for the final parameter
*
ints        movea.w  $112.w,a2        Vector to get integers
            jsr      (a2)             CA.GTINT
            bne.s    bad_exit
            move.w   0(a1,a6.l),d0    Use this RI space later
            swap     d7               D7 is element count
            move.w   d5,d4            Work out start offset
            ext.l    d4               Avoid +/- 32768 quirks
            add.l    d4,d4            Scale D4 for integers
            adda.l   d4,a4            Advance to first element
search_int  cmp.w    d5,d7
            bcs.s    finished
            cmp.w    0(a4,a6.l),d0    Extract array element
            beq.s    stack_int        Return D5 if D0 matches
            addq.w   #1,d5
            addq.l   #2,a4            Advance to next element
            bra.s    search_int
*
* Search a table of floating-point array elements
*
floats      movea.w  $114.w,a2        Vector to get floats
            jsr      (a2)             CA.GTFP
            bne.s    bad_exit
            move.w   0(a1,a6.l),d0    Fetch exponent
            move.l   2(a1,a6.l),d1    Fetch mantissa
            addq.l   #4,$58(a6)       BV.RIP for integer result
            swap     d7               D7 is element count
            ext.l    d5
            move.l   d5,d4
            add.l    d4,d4            Scale D4 by 6 for floats
            add.l    d5,d4            D4 := D5 * 3
            add.l    d4,d4            D4 := D5 * 6
            adda.l   d4,a4            Advance to first element
search_fp   cmp.w    d5,d7
            bcs.s    finished
            cmp.w    0(a4,a6.l),d0    Compare exponents
            bne.s    next_float
            cmp.l    2(a4,a6.l),d1    Compare mantissae
            beq.s    stack_int
next_float  addq.w   #1,d5
            addq.l   #6,a4            Step to next value
            bra.s    search_fp
*
* Search string array elements for the last parameter
*
strings     movea.w  $116.w,a2        Vector to get strings
            jsr      (a2)             CA.GTSTR
            bne.s    exit
            moveq    #1,d0            Allow for the odd byte
            add.w    0(a1,a6.l),d0    Unstack the text length
            bclr     #0,d0            D0 is length rounded up
            add.l    d0,$58(a6)       Leave one word for result
            move.w   d7,d6            D6 is string stride
            move.w   d5,d4            Work out start offset
            mulu     d6,d4            Scale for elements
            adda.l   d4,a4            Advance to first element
            movea.l  a4,a0            Set A0 for UT.CSTR
            swap     d7               Now D7 is element count
            movea.w  $E6.w,a4         Pick up UT.CSTR vector
search_str  cmp.w    d5,d7            Are we past the end?
            bcs.s    finished
            moveq    #3,d0            Sinclair comparison type
            jsr      (a4)             Call UT.CSTR
            beq.s    stack_int        Return D5 if they match
            addq.w   #1,d5
            adda.w   d6,a0            Next; Note implicit EXT.L
            bra.s    search_str
*
* Return location 0-32767, or -7 if the value was not found
*
finished    moveq    #-7,d5           NOT FOUND Qdos error code
stack_int   move.l   $58(a6),a1       Retrieve BV.RIP
            move.w   d5,0(a1,a6.l)    Stack a one word reply
            moveq    #3,d4            Indicate integer result
            moveq    #0,d0            Signal ERR.OK, it worked
exit        rts
*
define      dc.w     0,0              No procedures
            dc.w     2                One long-named function
            dc.w     inarray-*
            dc.b     8,'INARRAY%'
            dc.w     0
            end
```

SuperBasic, or 136 seconds for Minerva 1.64 with all the stops pulled out to boost integer arithmetic.

String searches are relatively slow, although still much faster than an interpreted loop, because INARRAY% uses the standard rom routine UT.CSTR to compare string values. This allows four types of comparison, so it can match strings which contain slight differences, but which a human might consider equivalent.

Differences of case may be ignored, so 'CAPS' matches 'Caps' and 'capS'. The value of imbedded numbers can be taken into account, so '7' matches '007' and '7.0'. By default INARRAY% uses the most sophisticated 'type 3' string comparisons, but you can select other types with a single POKE, discussed later.

Numeric searches run at much the same speed regardless of the values involved, but string searches are much more dependent on the text being compared. At best, UT.CSTR can compare about 3300 strings per second on an unexpanded QL, but this falls to 500 per second, or even less, if there are many close matches or strings of digits in the array.

INARRAY% is the equivalent of a loop using the == operator to compare values, rather than INSTR. String comparison stops as soon as a mis-match is detected, and INARRAY% steps on to check the next element.

INARRAY% returns a single integer value, so it can scan a maximum of 32768 array elements, returning a value between 0 and 32767. If the last element is reached, and no match is found, INARRAY% returns -7, the error code which signifies 'not found'.

Like MAXIMUM and MINIMUM, introduced last month, INARRAY% needs access to the array descriptors that record Basic array dimensions. It follows that INARRAY% is not suitable for use in *Turbo* or *Supercharged* tasks, as those compilers do not allow access to array descriptors.

INARRAY% has been tested successfully with interpreted SuperBasic from Sinclair's early AH roms to the late Sigma FP (derived from MG roms) and Minerva 1.64. It also works fine in tasks compiled with *QLiberator*, as these use the same variable allocation scheme as Sinclair's interpreter.

INARRAY% is simple to use. It accepts two or three parameters. The first parameter is the name of the array to be scanned. This may be a two-dimensional array of characters, declared with a line like DIM a$(640,80), or any shape of numeric array, from one to fifteen dimensions. The second parameter is an optional integer, indicating the number of the first element to be compared. This defaults to zero, so the array is scanned from the start; higher values allow elements to be skipped, so that every match in an array can be found with a succession of calls to INARRAY%.

The last parameter is the value to be located, which may be a constant, variable or expression. It is coerced to match the type of the array being searched. The following example prints the location of every instance of the value PI in array NUMS:

```
LET p%=0
REPeat scan
  p%=INARRAY%(nums,p%,PI)
  IF p%<0 : EXIT scan
  PRINT "Match at ";p%
END REPeat scan
```

If INARRAY% finds the required value, it returns an integer corresponding to the number of the matching element in the array. This is simple enough in the case of a one-dimensional array, but a little more complicated if the array has more dimensions. You need to use DIV and MOD to convert the integer result into values for each individual subscript.

SuperBasic stores arrays in memory with the last subscript changing most quickly. Subscripts start at zero, so DIM Z%(50,100) reserves space for 51*101 = 5151 elements, numbered from 0 for Z%(0,0) via 1, element Z%(0,1), to number 5150, for the final element Z%(50,100).

To locate an element in a two-dimensional array, you need to know the 'stride' between successive values of the first subscript - in other words, the number of elements in each row of the array. Z% has 101 elements in each row, as values for the last subscript vary from 0 to 100.

If POS% is the position of a match in Z%, returned by INARRAY%, POS% DIV 101 gives the first subscript of the matching element and POS% MOD 101 gives the second subscript.

The same principle holds for arrays of more than two dimensions. It is not difficult to work out the sub scripts using successive applications of MOD and DIV for each extra dimension. You should not go wrong as long as you work your way from right to left; remember to count element zero, allowing each slice one more element than the corresponding number in the DIM statement.

The most likely error report from INARRAY% is 'bad parameter', returned if the first parameter is absent or not an array. The same message crops up if the array has more than 32768 elements, if a string array has the wrong number of dimensions, or the optional parameter has a negative value.

INARRAY% attempts to coerce the value of the last parameter to match the type of the array. Reports like 'overflow' or 'error in expression' appear if the value is not suitable, as in cases like INARRAY%(TABLE,"") or INAR
RAY%(Z%,1E6).

As usual, the code for INARRAY% is listed in two forms. **Listing one** is the annotated Devpac assembly code. You can type this into your own assembler if you wish to develop or modify the routine.

Listing two uses the standard DIY Toolkit hex loader to create a short file containing the binary code for INARRAY%. To use this, type in and RUN the SuperBasic, then specify a name for the code file, such as INARRAY_CODE. Once the code file has been generated, load it with LRESPR from Toolkit 2 or this sequence of standard QL commands:

```
x=RESPR(348)
LBYTES "FLP1_INARRAY_CODE",x
CALL x
```

You may wish to change the device or file name in the second line to suit your system. If you do not wish to retype and check the listings you can obtain DIY routines on disk. Complete assembler source, binary code and documentation for INARRAY% and the MAXIMUM and MINIMUM routines presented last month is available by post from DIY Toolkit, Cwm Gwen Hall, Pencader, Dyfed SA39 9HA. Send £7 for DIY Toolkit Volume Z, one of 19 volumes of programs, text and examples derived from the *QL World* series. Alternatively call Richard Alexander on 0559 384574 or send a stamped self-addressed envelope for details of the volumes.

Please note that the programs, documentation and listings in DIY Toolkit are protected by UK copyright law and may not be reproduced in any way without the permission of the author. They are not in the public domain. The only exception to this rule is that QL programmers are permitted to supply individual files of DIY Toolkit code with application programs that need DIY extensions in order to work. Please do not abuse this concession.

The structure of SuperBasic array descriptors was introduced last month. The new code shows how multi-dimensional arrays can be scanned, and demonstrates the use of the string comparison vector UT.CSTR.

After the usual short initialisation routine, INARRAY% starts by checking and processing the array parameter. This has the value 3, signifying an array, at the start of the Name Table entry addressed by (A3,A6), followed by the data type: 1 for strings, 2 for floating-point values and 3 for integers. The special case of sub-strings, with type 0, is excluded. Next, the

Listing two

```
100 REMark Sinclair QL World HEX LOADER v 3
110 REMark by Marcus Jeffery & Simon N Goodwin
120 :
150 CLS: RESTORE : READ space: start=RESPR(space)
160 PRINT "Loading Hex..." : HEX_LOAD start
170 INPUT "Save to file...";f$
180 SBYTES f$,start,byte : STOP
190 :
200 DEFine FuNction DECIMAL(x)
210 RETurn CODE(h$(x))-48-7*(h$(x)>"9")
220 END DEFine DECIMAL
230 :
240 DEFine PROCedure HEX_LOAD(start)
290 byte = 0 : checksum = 0
300 REPeat load_hex_digits
310    READ h$
320    IF h$="*" : EXIT load_hex_digits
330    IF LEN(h$) MOD 2
340       PRINT"Odd number of hex digits in: ";h$
350       STOP
360    END IF
370    FOR b = 1 TO LEN(h$) STEP 2
380       hb = DECIMAL(b) : lb = DECIMAL(b+1)
390       IF hb<0 OR hb>15 OR lb<0 OR lb>15
400          PRINT"Illegal hex digit in: ";h$ : STOP
420       END IF
430       POKE start+byte,16*hb+lb
440       checksum = checksum + 16*hb + lb
450       byte = byte + 1
460    END FOR b
470 END REPeat load_hex_digits
480 READ check
490 IF check <> checksum
500    PRINT"Checksum incorrect. Recheck data.":STOP
520 END IF
530 PRINT"Checksum correct, data entered at: ";start
560 END DEFine HEX_LOAD
570 :
580 REMark Space requirements for the machine code
590 DATA 348
600 :
610 REMark Machine code data
640 DATA "43FA01463478011Ø","4ED2BBCB67780C33"
650 DATA "ØØØ3E80066707A03","CA33E80167682273"
660 DATA "E804286E0028D3CC","D9F1E8003031E804"
670 DATA "7E010C050001661Ø","5540664A3E31E806"
680 DATA "3A31E8086A1A603E","3431E8065242CEC2"
690 DATA "5889534066F25387","BEBCØØØØ7FFF6226"
700 DATA "508BBBCB67204847","3E057A0045EB0008"
710 DATA "BBCA67282A0D2A4A","3478011124E92608"
720 DATA "3031E8006A0470F1","4E7554AE0058264D"
730 DATA "2A453A0045EB0008","BBCA66EA0C470003"
740 DATA "6B28665E34780112","4E9266DC3031E800"
750 DATA "4847380548C4D884","D9C4BE45657AB074"
760 DATA "E80067765245548C","60F03478011144E92"
770 DATA "66B63031E8002231","E80258AE00584847"
780 DATA "48C52805D884D885","D884D9C4BE456548"
790 DATA "B074E8006606B2B4","E802673E52455C8C"
800 DATA "60EA347801164E92","663C7001D071E800"
810 DATA "0880000D1AE0058","3C073805C8C6D9C4"
820 DATA "204C4847387800E6","BE45650C70034E94"
830 DATA "67085245D0C660F0","7AF9226E00583385"
840 DATA "E80078037000 4E75","0000000000002FEBC"
850 DATA "08494E4152524159","25000000","*",32833
```

code points A1 at the array descriptor in the Variable Values area and puts the offset of the first element into A4.

If a string array is being processed, INARRAY% checks for two dimensions and picks up the stride between elements from the end of the descriptor. The type in D5 is replaced by the stride, which will always be four bytes or more, as every string element consists of a two byte length and two or more bytes for the text. The length must be stored at an even address, so DIM rounds up string lengths to the next even number, so DIM T$(3) allocates four bytes for text, plus two for the length.

In the case of numeric values the type is unchanged and the loop at GET_INDEX scans each subscript in turn, working out the total number of elements in D7. This value is transfered to the high word of D7, so the type can fit in the lower word. This shuffling is necessary because subsequent parameter-fetching may corrupt all data registers except D5 and D7, and we need D5 to store the number of the first element to be scanned.

If A3 + 8 = A5, only one parameter remains, so D5 is zero, indicating that we should scan from the start. Otherwise CA.GTINT is used to fetch the next parameter and copy its value from the RI stack to D5. I take care to adjust BV.RIP, the RI Stack pointer at $58(A6), once the value has been read; otherwise the stack grows by two bytes every time the optional parameter was used, slowly gobbling memory and causing strange results if other values were already waiting on the stack.

By the time we reach the label PATTERN there should be just one parameter left - the value we hope to find. Now INARRAY% forks into three distinct routines, one for each data type.

The INTS routine fetches an integer with CA.GTINT, leaving its space on the Maths Stack for the result, then scans an integer array. D5.W holds the current element number, and D7 is swapped to retrieve the total number of elements. D5 is incremented after each unsuccessful comparison, until it reaches D7 or a match is found.

FLOATS works similarly, using CA.GTFP to read a six byte floating point value for comparison. ADDQ.L #4,$58(A6) adjusts the Maths Stack pointer to leave two bytes of space for the integer result. Three ADD.L instructions are used to multiply the initial element number by six, advancing A4 through the array to the required element.

The floating-point comparisons are almost as fast as integer ones, as the SEARCH_FP loop starts by comparing the exponent words of the element and the pattern; only if these match does it go on to compare the mantissae in the following four bytes.

The scanner labelled STRINGS fetches a parameter with CA.GTSTR, then adjusts BV.RIP to leave room on the maths stack for the integer result. A1 is left pointing at the parameter length, and A0 is made to point at the first element to bee scanned, to suit UT.CSTR. Just before the loop at SEARCH_STR the code moves the UT.CSTR vector into A4, to save fetching it before every comparison.

UT.CSTR is unusual because it does not return an error code in D0; instead it returns zero if the strings at (A0,A6) and (A1,A6) match, -1 if the A0 string comes first, and +1 otherwise. The loop stops as soon as BEQ.S detects a match.

All three routines return via STACK_INT, with the element number in D5, if a match is found. Otherwise they branch to FINISHED and return -7, the Qdos code signifying 'not found'.

One advantage of using the rom routines to compare strings is that the type of comparison can easily be changed. By default INARRAY% uses type 3 comparisons, taking special account of imbedded numbers and ignoring the case of letters. The alternatives are Type 0 comparisons, which require an exact match, type 1, which only ignores letter case, and type 2, which matches equivalent numbers but distiguishes between small letters and capitals.

A simple POKE sets the type of string comparison performed by INARRAY%. This can most easily be set immediately after the code has been loaded into memory. If X is the address of the start of the code, X+301 holds the comparison type, so PEEK(X+301) should give 3 initially. POKE X+301,0 for strict comparisons, or POKE 1 or 2 for the other possibilities.

You can change the comparison type at any time, as long as you keep track of the address to POKE; this is an obvious application for SET, from QL World April and May 1991, or DIY Toolkit disk Volume U.

IF AN ADVERT IS WRONG, WHO PUTS IT RIGHT?

We do.

The Advertising Standards Authority ensures advertisements meet with the strict Code of Advertising Practice. So if you question an advertiser, they have to answer to us.

To find out more about the ASA, please write to Advertising Standards Authority, Dept. X, Brook House, Torrington Place, London WC1E 7HN.

ASA

This space is donated in the interests of high standards in advertisements.

TF Services

MINERVA

The ULTIMATE operating system upgrade

MKII MINERVA now with battery for 256 bytes ram, CRASHPROOF clock & interface (I²C) for robots etc. Can autoboot from battery ram.

Other features (common to MKI/MKII): Autoboot on reset/ Multiple Basic/ Faster scheduler-graphics(10% of Lightning)-string handling etc/ WHEN ERROR/ Second screen/TRACE Foreign kybd drivers/ 'Warm' fast reset etc

MKI to MKI UPGRADES: up to 1.79 - £5. Rest FREE (send rom (PROTECT LEGS!) and disk or 2 mdvs)

Original MKI..... £40 MKII.....£65

ALL VERSIONS FULLY GOLD CARD COMPATIBLE

HERMES

A replacement QL co-processor (IC 8049 IPC)

Do you get keyboard bounce?
Do you find fast serial input unreliable?
Do you want to run an HST 14400 bps modem?

If you can say YES to any one of these, then this is a product you have been waiting for

- Gives up to 19200 bps RELIABLE serial input
- Independent baud rates
- Stops keyboard 'bounce' (unwanted repeat chrs)
- Improves 'fuzzy' and 'random' sound
- Provides extra input and output lines
- Key click
- Reset/INT7 invoked more safely

FITTING is a simple job. Simply remove the QL top (8 screws) and replace the chip marked '8049' next to microdrive 1.

PRICE (including manual & software) £25

BULLETIN BOARD

3885 - The first UK QL scrolling bulletin board. Now even larger with international ECHOmail and 10mb of files (including the WHOLE King James' bible!)

071-706 2379 V21/23/22/22bis 24hrs

QL SPARES

Faulty QL boards (excl plug in ICs).....	£9		
Keyboard membrane...	£8	Circuit diags...	£2
68008 cpu.....	£8	1377 PAL.....	£3
JM rom set.....	£10	Power supply...	£12
8302 ULA.....	£10	8301 ULA.....	£10
8049.....	£8	MDV ULA	£12

Other components (sockets etc)... plse PHONE

QualSoft

Viewdata/VT52/VT100 QL TERMINAL EMULATOR

Multitasking for electronic mail, PRESTEL etc, CET download. Compatible with Tandata Qconnect QL modem/modaptor. Use Qconnect RS232C to drive unbuffered modems. Comprehensive A5 manual.

Software (3.5" or mdv) & manual £30

FILE TRANSFER - to transfer text AND programs error free CHOCCIEMD between computers

Available for IBM compats, ATARI ST & Sinclair QL. Connects to Psion range with RS232C comms link. Software £7.50 per m/c. Cable £10 (name 2 m/cs)

QL REPAIRS

QLs tested with Thorn EMI test rig and ROM software (MDV hardware extra if required)

£27 - 6 month guarantee

All prices include VAT and postage in the UK

MAIL ORDER ONLY

VISA

12 Bouverie Place, London N2 1RB (tel: 071-724 9053)

SCROLLING BBS FOR ORDERING/QL HELP and FAX - 071-706 2379

Just for Starters

In part 2 of his SuperBase tutorial, Don Smith continues adding procedures. First, the number of fields must be entered to keep the database flexible.

For those who tried out the simple outline database in the February 1992 issue, we will now increase the number of fields before adding the missing procedures. Assuming that you entered 30 items, that item$ was a book title and num the price, and that you entered file$ as 'library' then you have one **file**, called library, made up of 30 **records**, each record describing one book and each comprising two **fields**.

It is too limiting to have only two fields; however, while anything can be written on a file card, there are some constraints when inputting data into variables. There are three kinds of variables:

1) Non-numeric string (nns) variables, like item$, always ending in $. These will accept any printable keyboard character

2) Floating point (fp) variables, like num or x, which will hold gigantic numbers, but anything over 999999 uses an exponential configuration, for example: 55 million = 5.5E7. This is unhelpful for telephone numbers, and for these an nns variable of suitable length would be used.

3) Binary variable, such as quan%, which will hold +/-32767, but no decimal points.

A suitable analogy of how data is stored, but not how the computer stores it, is to start with a pile of small, open boxes. Considering num(max) in line 470 of the listings, if max was 100, imagine 101 boxes in a straight line. All automatically contain 0 at the start, and the boxes are numbered 0 to 100. Your data starts be going into box 1, so I have used box 0 to hold the total number of entries (see lines 450, 600, 730). This is a one-dimensional array, and each box uses up 5 bytes. A binary variable array would be no different, except that each box would use up only 2 bytes.

For item$(100,30), consider a rectangle of boxes, 31 wide and 101 deep: that is, two-dimensional. Not only is there a zero column downwards but also a zero row at the top. The top left-hand corner would be box 0,0, next to it 0,1, and so on up to 0,30. Sorting often uses this zero row. The next row down commences with box 1,0, then 1,1, and so on up to 1,30, but box 1,0 and all the other zero boxes below must not be used, since the QL stores the true length of the stored variable there. If your first entry was 'The 1990 Almanack', item$(1,0) will contain the number 17.

Enough of explanations for now. Hope-

```
185  print\to 10,'I'to 15;'Sort'to 50;'J'to 55;'Abstract'

255   =105: ontop=cost(0)+1: aut$(ontop)='Zzz z': title$(ontop)
      ='Zzz z': year%(ontop)=9999: cost(ontop)=99999: sort

257   =106: abstract

430  dim aut$(max,25), title$(max,40),year%(max),cost(max)
```

Listing one: Four lines to be added to February's listing (see text)

```
580  aut$(x)=nam$: print x;fill$('.',5-len(x));aut$(x);
     fill$('.',26-len(aut$(x)));
590  cls#0: print#0,'Title'\fill$('-',40):input#0,title$(x):
     print title$(x);fill$('.',41-len(title$(x)));
593  cls#0: print#0,'Year'\fill$('-',4): input#0,year%(x):
     print year%(x);fill$('.',5-len(year%(x)));
596  cls#0: input#0,'Cost'to 10;cost(x): print cost(x)
600  cost(0)=x
670  print x;fill$('.',5-len(x));aut$(x);fill$('.',26-len(aut$(x)));
     title$(x);fill$('.',41-len(title$(x)));year%(x);
     fill$('.',5-len(year%(x)));cost(x)
```

Listing two

fully you have coped with editing, and corrected any errors. I have just wasted a full 10 minutes wondering why one of the following procedures wouldn't work - finally realising that I had missed off a $ sign! With both cartridges in place, enter lrun mdv1_ your name. Remember that Ctrl-Spacebar is like an emergency brake pedal, there to press when things get out of control, allowing time to stop and make a fresh start. We need an extra line in the menu, so list the program and stop it with Ctrl-F5 then Ctrl-Space to free the cursor. Type the following (line 185 of the listing), including line number, but don't enter it.

```
185 print\to 10,'I'to 15;'Sort'to 50;'J' to
      55;'Abstract'
```

Now for a useful short cut. Press the right-hand down arrow: in goes line 185, down comes 190. Alter I to K, enter, and type in line 255 of the listing:

```
255 =105: ontop=cost(0)+1: aut$(ontop)=
      Zzz z': title$(ontop)='Zzz z':
      year%(ontop)=9999:
      cost(ontop)=99999: sort
```

Press the down arrow, alter =105 to =107, enter, and type in line 257:

```
257 =106:abstract
```

Enter and continue listing until line 300 is at the top, then free the cursor. We shall now tailor the database to a library file with four fields: Author, Title, Year and Cost. Type in the following (line 430) but don't enter it:

```
430 dimaut$(max,25), title$(max,40),yea
      r%(max),cost(max)
```

Now for another short cut. Press first the down, then the up arrows. The line goes into the program and reappears down below again. Alter 430 to 470, producing two identical lines only typed once. Edit 440 by deleting the last two items and replace them with aut$(x), title$(x), year%(x), cost(x). Edit 450 from num to cost, and on to the records procedure. Enter CLS, enter LIST 500 to - stop the screen when full. Edit where necessary, or enter both line number and line. A symbol ^ indicates a blank space. Edit 540 and 550 from num to cost. Alter Name in line 560 to Author, and 30 in fill$ to 25. (see listing two)

Lines 730/740 alter num(0) to cost(0), line 740 removes item$ and num and replaces them with the four fields as in line 440 (not forgetting the backslash), and num(0) to cost(0). Line 830 alter num to cost. No more alterations now, but the next few procedures may cause problems since they concern the printer. I bought an Epson FX-8C simply because the QL is Epson-based. It seemed logical and it accepts every Epson manual command. Unfortunately, I cannot offer advice to owners of other machines, and can only provide the following commands for printing within this program **(see listing three).**

```
1020  :
1030  defproc del: print#4,chr$(27);'!';chr$(17);: enddef
1040  defproc pica: print#4,chr$(27);'!';chr$(0);: enddef
1050  defproc con: print#4,chr$(27);'!';chr$(4);: enddef
1060  defproc elite: print#4,chr$(27);'!';chr$(1);: enddef
1070  defproc empica: print#4,chr$(27);'!';chr$(8);: enddef
1080  defproc und: print#4,chr$(27);'-';chr$(1);: enddef
1090  defproc cancelund: print#4,chr$(27);'-';chr$(0);: enddef
1100  defproc american: print#4,chr$(27);'R';chr$(0);: enddef
1110  defproc linefeed: print#4,chr$(10);: enddef
1120  defproc formfeed: print#4,chr$(12);: enddef
1130  defproc space10: print#4,chr$(27);'A';chr$(10);: enddef
1140  defproc space12: print#4,chr$(27);'A';chr$(12);: enddef
```

Listing three

```
1150  :
1160  rem------SORT MENU----------
1170  defproc sort
1180  print\\\\to 35;'SLOW SORT'\\to 15;'A'to 20;'Author'to 55;
      'B'to 60;'Title'
1190  print\to 15;'C'to 20;'Year'to 55;'D'to 60;'Cost'
1200  print\\to 35;'QUICK SORT'\\to 15;'E'to 20;'Author'to 55;
      'F'to 60;'Title'
1210  print\to 15;'G'to 20;'Year'to 55;'H'to 60;'Cost'
1220  line 30,50 to 30,15 to 150,15 to 150,50 to 30,50 to 30,85
      to 150,85 to 150,50
1230  j=code(inkey$(-1)): cls
1240  sel on j
1250   =97: slowaut: =98: slowtitle
1260   =99: slowyr: =100: slowco
1270   =101: final=cost(0): fastaut 1,final: =102: final=cost(0):
        fastitle 1,final
1280   =103: final=cost(0): fastyr 1,final: =104: final=cost(0):
        fasco 1,final
1290  end sel
1295  menu
1300  enddef
```

Listing four

```
1310  :
1320  rem------SLOW SORT FOR AUTHORS----------
1330  defproc slowaut
1340  cls: e=cost(0)
1350  for f=2 to e
1360   at 14,30: print f
1370   g=f
1380   if aut$(g)>=aut$(g-1): goto 1440: else: aut$(0)=aut$(g):
       title$(0)=title$(g): year%(0)=year%(g): cost(0)=cost(g)
1390   rep sortloop
1400    if aut$(0)<aut$(g-1): aut$(g)=aut$(g-1): title$(g)=
        title$(g-1): year%(g)=year%(g-1): cost(g)=cost(g-1):
        else: exit sortloop
1410    if g>1: g=g-1: else: exit sortloop
1420   endrep sortloop
1430   aut$(g)=aut$(0): title$(g)=title$(0): year%(g)=year%(0):
       cost(g)=cost(0)
1440  endfor f
1450  cost(0)=e
1460  menu
1470  enddef
```

Listing five

Del gives a much blacker, double-strike elite (which takes twice as long to print!)

```
1480  :
1490  rem------FAST SORT FOR AUTHORS----------
1500  defproc fastaut(lower,upper)
1510  local partel
1520  if lower<upper then
1530   partel=upper+1
1540   partaut lower,partel
1550   fastaut lower,partel-1
1560   fastaut partel+1,upper
1570  endif
1580  enddef
1590  :
1600  defproc partaut(i,j)
1610  local k,valaut$,valtitle$,valyear%,valcost
1620  valaut$=aut$(i): valtitle$=title$(i): valyear%=year%(i):
      valcost=cost (i)
1630  k=i
1640  rep autloop
1650   rep inck: k=k+1: at 15,40: print k: if aut$(k)>=valaut$
       exit inck
1660   rep decj: j=j-1: if aut$(j)<=valaut$: exit decj
1670   if k<j: swapaut k,j: else: exit autloop
1680  endrep autloop
1690  aut$(i)=aut$(j): title$(i)=title$(j): year%(i)=year%(j):
      cost(i)=cost(j)
1700  aut$(j)=valaut$: title$(j)=valtitle$: year%(j)=valyear%:
      cost(j)=valcost
1710  enddef
1720:
1730  defproc swapaut(a,b)
1740  local temaut$,temtitle$,temyear%,temcost
1750  temaut$=aut$(a): temtitle$=title$(a): temyear%=year%(a):
      temcost=cost(a)
1760  aut$(a)=aut$(b): title$(a)=title$(b): year%(a)=year%(b):
      cost(a)=cost(b)
1770  aut$(b)=temaut$: title$(b)=temtitle$: year%(b)=temyear%:
      cost(b)=temcost
1780  enddef
```

Listing six

```
1790  :
1800  rem------PRINT OUT DATA----------
1810  defproc prin
1820  elite
1830  for x=1 to cost(0)
1840   print#4,x;fill$('.',6-len(x));aut$(x);fill$('.',26-
       len(aut$(x)));title$(x);fill$('.',41-len(title$(x)));
       year%(x);fill$('.',5-len(year%(x)));cost(x)
1850  endfor x
1860  formfeed
1870  menu
1880  enddef


1890  :
1900  rem------PRINT OUT PROGRAM----------
1910  defproc listprog
1920  empica: und: print#4,'LIBRARY COLLECTION': cancelund
1930  linefeed: elite: american: space10: list#4
1940  formfeed
1950  menu
1960  enddef



1970  :
1980  rem------COMPUTER MEMORY----------
1990  defproc room
2000  cls#0: print#0,int((PEEK_L(163856)-PEEK_L(163852))/1000);
      'K left',int((PEEK_L(163872)-PEEK_L(163856))/1000);'K used'
2010  enddef
```

Listing seven

and con is condensed which, while theoretically giving 137 characters per line, is usually constrained to 132. Empica is emphasised pica; und is underline, and american ensures that # is printed and not £. Default line spacing is one-sixth of an inch apart (12/72nds) and space10 closes this down to 10/72nds. This gives 72 lines per page, instead of 60. Elite allows 97 characters per line, pica 80. Many of these are used when printing the program. We now come to a big block of lines for sorting. The sort menu allows for either slow or quick sorts of all four fields, but only those for author are given. **(see listing six).**

The quicksort is based entirely on Marcus Jeffery's article way back in August 1985 and, as far as I am aware, has not been bettered since. I compared both the above sorts on identical random numbers with the following (partial) results:

Slow:quick - 50 items = 15secs:5secs, 100 = 46:10, 500 = over 2000:68, 1000 = ??:150, and 16 minutes for 5000 using the quicksort.

Slow would be measured in hours. The quicksort, however, has two foibles. The first requires that arrays are not completely filled so, when loading, overestimate total entries, otherwise line 255 is inoperable. The second quirk is encountered when adding a small number of additional entries to a large, sorted list. If you then quicksort, it becomes as slow as a bubble sort until it has worked its way down to the new entries. If Marcus reads this, I would dearly love to read his solution to the problem.

Another sort

One way out of the problem is to sort by some other field, then sort the field required so that both operate on unsorted fields. A quick way to program sorts for other fields, having typed the two examples for author, is as follows. Determine the last program line, in this case 2010. Enter auto 1310 (the first line of the author slowsort) and down comes line 1310. Alter 1310 to 2020, and enter. 1320 now appears, which is altered to 2030 and continues to line 1470. List along to 2020 and halt the screen. You will see two lines starting with 'if aut$', so change both pairs of aut$ to whichever other field you wish to sort. Also, if duplicating the quicksort, you will find two lines side by side with 'if aut$' and 'valaut$'. These want altering to, say, title$ and valtitle$ or year% and valyear%.

These above three procedures conclude this article. Alter and abstract are still to come. The final three lines are important since, if you find yourself with only 20K or so left, it is time to fit an Expanderam. You can build 'room' into the program, perhaps after saving, or access it from the keyboard.

NOTICEBOARD

This is where I fill in some more detail about QL World's change of venue.

Many people kindly wrote to congratulate us on having survived Robert Maxwell's demise and acquiring a new publisher. The pleasure of your letters was not diminished by the knowledge that our new publisher was unlikely to be the publisher everyone thought it was. I only wished we could have said more, sooner. The April issue was concluded nearly two months ago, at which time we didn't have sufficient information to know what was happening.

All this was compounded by two other factors - one: our previous now-management-boughtout owners, HHL Publishing, were in the process of removing their operation from the Maxwell-owned Panini House (where QL World's management was based) to their central office at Greater London House; two: our new publishers, Arcwind Publications, is in the process of relocating to better offices. Just two small symptoms of the curious state of the UK property market at present.

It appears to be the closure of another Maxwell-owned company based at Panini House that led to the rumour of QL World's demise - but, as you can see, it's not so. I did try to keep Phil Borman at Quanta informed as I had planned - but by the time we were able to tell him what was happening, we were running even later than expected. Quanta however would have been a more reliable source of information than our ex-office-receptionist. Regrettably I was unable to do more than drop a hint about that.

Publisher Mark Kasprowicz, writing on page 3 of this issue, has mentioned the extention of subscriptions to take into account that we are only publishing one issue for the May/June 1992 period. This was a near miss. We were aiming to get the May issue out, albeit late, but the extra strain imposed by all the moving (simply tracing our post has become a challenge in its own right) brought us to the point that we would be 'catching up' for months if we tried to do two issues in quick succession.

QL World's postal address is now:

QL World
Arcwind Publications
c/o The Coach House
Medcroft Road
Tackley
Oxon OX5 3AH

Post sent to this address will still reach our office promptly even when we have a new office address, so fear not.

Important note: Arcwind have assumed ownership of QL World from the May 1992 issue onwards. Any transactions involving the April issue or earlier - author's payments, missing issues, etc., - are still the domain of HHL Publishing, who can be contacted at 071 388 3171. Michelle Fitzgerald will still have access to QL World records for the issued published by MCPC and HHL.

Various people have enquired about our attitude to yachting since Robert Maxwell declined. We think this makes it doubly significant that Arcwind started its publishing history several years ago with a journal called Windsurf - a rather more down-to-earth pastime than luxury-yachting - which is still going strong.

I now have some new notepaper, albeit of a temporary nature, so I will be able to get letters out to people who need them. The back issue situation will be a bit uncertain until we know what we have possession of and where it's going to be stored.

Mark and Jim from Arcwind went to the Bristol Quanta Sub-group workshop at Clevedon at the end of April and met many of the QL traders and local members. The meeting was a great success, with traders from the UK and Germany (and Wales, of course). It was a pity that the hole in our schedule left us unable to print the details of the Meeting in time - but look on the bright side: we didn't run the poem, either!

Mark also remarked on page 3 that we are far from the elevated towers of Central London. Only country dwellers will appreciate just how far, but there are advantages. Quite apart from watching my own pear tree trying to produce its quota of two fruit each summer, I can now while away my periodic trips to Arcwind in admiring the vast expanse of oilseed flowering around here. It looks as though somebody left the lights on in Lothlorien - necessary now that a change in QL World's colour imposition has (temporarily, I hope) curbed my excesses on the Open Channel pages. But then, I don't suffer from hay fever. We should certainly have no more trouble with the postal service here than we had in London EC1. Not until the snowdrifts start, anyway.

No more concrete information at present, except that I hope to be able shortly to give sensible answers to some people who have been waiting for them.

See you next month.

The Editor.

In the second part of his assessment of desktop publishing for the QL, Geoff Wicks looks at fonts.

A Question of Dots

♦ Part 2 ♦

Desktop publishing programs, and especially good desktop publishing programs, are not cheap. For this reason I doubt if many people buy such programs on impulse. I also suspect that most buyers are fairly busy people running their own business, or are actively involved in running some type of club or association. The temptation is to plunge straight in and use the program immediately. It is a temptation to avoid. One of the reasons that desktop publishing software is so expensive is that it is complicated, and as with all complicated programs there is a fairly long learning curve before you can get the best out of them.

My main reason for going into desktop publishing was that as secretary of the Workers' Council in the agency where I was employed, I wanted council publicity to stand out from other papers, reports and notices which were circulated. The agency was just getting into computers, and in a friendly rivalry with the 'computer expert' of the company I was trying to keep one step ahead of him. I knew I had an advantage in that he believed that the agency did not need to buy desktop publishing facilities, especially as that would mean extra training for the staff.

All the same, until Digital Precision released Professional Publisher my opinion was that the use I could make of desktop publishing was mainly restricted to short documents. What I was not to know is that just at the point at which I received Professional Publisher and was thinking of widening my horizons, a conflict was to arise in the Workers' Council which resulted in my sudden resignation on a matter of principle. Suddenly I had enough time on my hands to explore the capabilities of Professional Publisher in greater detail than I had expected.

Bewildering

A disk containing a desktop publishing program contains a bewildering array of files. If my calculations are correct, my edition of *Front Page Extra* consists of 26 files containing more than 193K of code. *Desktop Publisher Special Edition* uses all but three of 918 disk sectors and has 35 files, while Professional Publisher has 107 files and has only 12 out of 1404 sectors on the disk free. In addition to the main program, there can be runtime programs or toolkits, auxiliary programs or printer drivers which may have to be loaded and used separately, standard QL fonts, high resolution fonts and clip-art.

The first priority is to separate these files into a usable order. I have divided my version of Professional Publisher over a number of disks. One set contains the main program, toolkits, etc.; the second all the standard QL fonts; the third the high resolution fonts; and the fourth the clip art. You can then add material from other sources. *Lightning* has more QL fonts; you may wish to add fonts from other desktop publishing programs, or the Quanta library, as well as any you have written yourself. There are also extra fonts and clip-art on the *Professional Toolbox* disk. The Professional Publisher disk has an auxiliary program for converting high resolution fonts from DP's earlier program, *Desk Top Publisher*, and these should be added to your Professional Publisher disk if you have both programs. Look around for what you can add to your clip art files (although you must be careful not to infringe anyone else's copyright). You may just find a use for those provocative screens that come with *Eye-Q* – maybe you do not want the equivalent of a page 3 girl in your publications, but the screen can always be cropped to give a smiling face. If you have succeeded in producing a screen containing your company's logo, this deserves a place on your clip art disk.

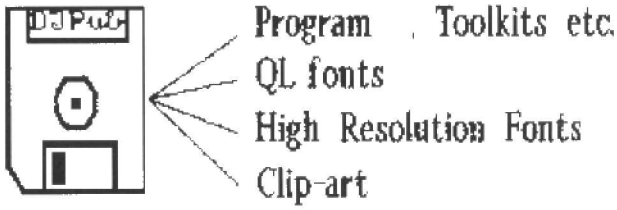
A few practical hints. When copying from the master disk, the WCOPY command on your disk interface is very useful. For example:

WCOPY flp1__qls to flp2__qls (note the double underscoring)

will copy all the Professional Publisher low resolution fonts from disk 1 to disk 2 (Digital Precision warn that with some disk interfaces this command cannot be used for copying clip-art files). Secondly, when your disk is complete it is helpful to make a hard copy of the directory. This is easily done using one line of Basic:

```
OPEN #3,ser1 : DIR #3,flp1_ : CLOSE #3
```

Finally, make at least one and preferably two backup copies of all your disks. You have spent hard-earned money on desktop publishing software, so do not skimp on building up a good system just for the few pounds that the extra disks will cost.



A DTP disc contains many types of files

The next stage is to make hard copy samples of all fonts, both low resolution and high resolution. You can of course just type out the lower and upper case alphabet and the numerals, but it is better to use the text importing routine to input two or three paragraphs of text, as this will give a better impression of what a font will look like if used in a document. It is also a good idea to headline each sample with the name of the font.

Experiment

If you are using Professional Publisher and the font is produced on a matrix larger than 16 x 16 make a printout of the fonts using both normal and condensed width/length printing. Even the most unlikely fonts can sometimes give an unexpected result when printed out using reduction techniques.

Your few paragraphs of text can be anything you like, but bear in mind you will be seeing the same text again and again. I chose the opening words of *Under Milk*

Wood which is the book I would probably take with me if marooned on a desert island. No matter how many times I see it, I never tire of the "sloeback, slow, black, crowblack, fishing boat bubbling sea".

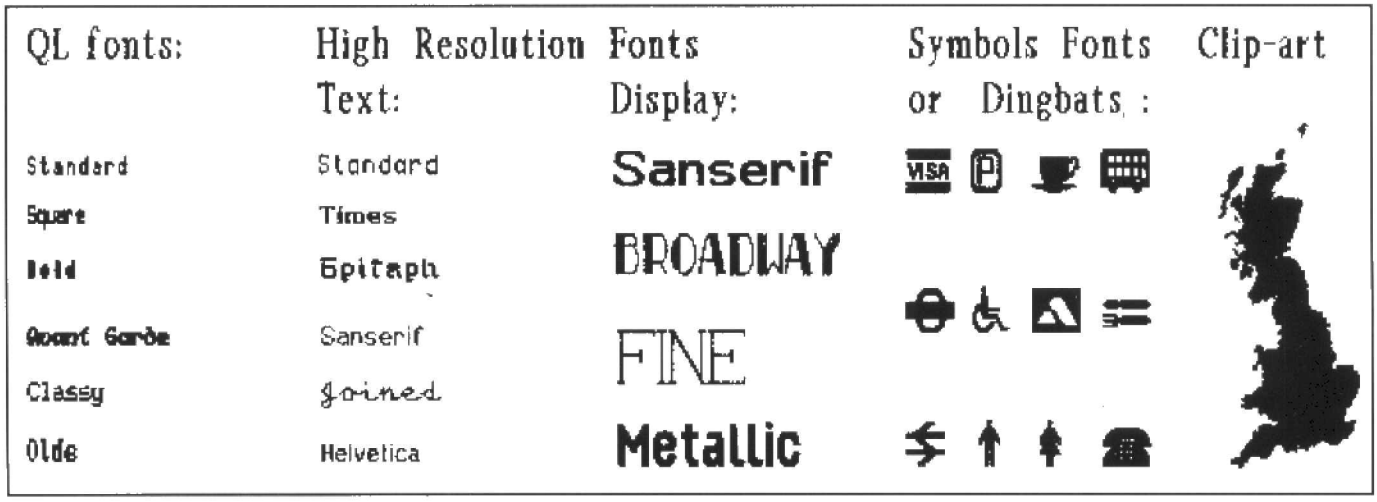
Not all fonts are suitable for printing out in this way. Some very decorative display fonts do not include both upper and lower case letters or numerals. For these fonts, just typing out the character set should be sufficient.

When you have completed your samples of the fonts, it is time to carry out a similar process with the clip-art files. A full size QL screen cannot be loaded in *Front Page* or *Desk Top Publisher* and you may find it easier to load the clip-art in Eye-Q or a similar program and print it out from there.

The process of printing out samples of the fonts and clip-art will place great demands on your time, energy, printer ribbon, paper and patience. You may begin to doubt your sanity for even thinking of going into desktop publishing, and start cursing Digital Precision for including 57 fonts and 11 clip art files with Professional Publisher, but persevere, as it will save you a lot of time later on, as well as bringing you great satisfaction.

When you have completed making all the hard copies, put them in a binder together with the printed out directories of your disks and the instruction manual for your program. Add a printout of any update information contained on the master disk or instructions for auxiliary programs such as *Grafix* and any other information you may find helpful. For example, my binder also includes the instruction manual for Eye-Q. This will now form an invaluable reference guide to which you will constantly refer when using your desktop publisher. When you are not sure what fonts to use or have forgotten the name of a font or a file, you will regret the time spent in compiling the binder. Furthermore, the font samples will be useful as demonstration material if you ever do work for third parties, and wish to show them the range.

We have now spent a lot of money on a desk top publisher. We have spent hours getting started on it, but a designed page has yet to see the light of day! I hope to redress the balance in my next article.



a CLUBS

FRANCE

QL Contact France. Contact: Jean-Louis Dianoux, 38-40 Rue Stephenson, 75018 Paris, France.

BELGIUM

Club Sinclair BruQsL (Belgium) Contact: Jaques Tasset, Aarlenstraat 104, 1040 Brussels, Belgium

SWEDEN

International QL Conference bulletin board system (Swedish and English). Contact: Michael Cronsten, System Operator, Jamten-TCL, S Soere 1073, 83030 Lit, Sweden.

USA

New England Sinclair QL User Group (USA) Membership Secretary: Sherm Waterman, PO Box 8763, Boston, MA 02114 8763, USA. Magazine: *NESQLUG News*. Editor: Peter Hale, 195 Central Ave., Chelsea, MA 02150, USA.

NORWAY

Norwegian All Sinclair Association (NASA) Contact: P Monstad, NASA, N-5580 Oelen, Norway. Magazine: *Sinclair Magazine*.

ITALY

Qitaly Club Chairman: Roberto Orlandi, Via Brescia 26, 25039 Travagliato (BS), Italy. Tel. (local) +39 30 6863311. Magazine: *Qitaly Magazine*. Editor: Dr Eros Forenzi, Via Valeriana 44, 23010 Berbenno (SO), Italy. Tel. (local) +39 342 492323.

TURKEY

QL Qlub (Turkey). Contact: Bulent Artuz, Prof. Sitesi B/1 D/5, Etiler 80600, Istanbul, Turkey.

AUSTRIA

DER Computer Club. Contact: Peter Postl, Stiegerg 5, 1150 Wien, Austria

ENGLAND

Quanta (UK) Membership Secretary, Bill Newell, 213 Manor Road, Benfleet, Essex SS7 4JD. Magazine: *Quanta*. Editor: Bill Fuggle, 20 Widnes Avenue, Selly Oak, Birmingham B29 6QE.

Bristol sub-group: Roy Brereton, 94 Teignmouth Rd, Clevedon, Avon BS21 6DR.

Essex sub-group: Dave Walker, 22 Kempton's Mead, Potters Bar, Herts EM6 3HZ.

London sub-group: Jeremy Davis, 6 Elmcroft Crescent, Harrow, Middlesex HA2 6HN.

Northern Ireland: Billy Turkington, Fairyhill, Rostrevor, Newry, Co., Down BT34 3BB.

Mid-Southern sub-group: Geoff Fish, 44 Billing Avenue, Wokingham, Berks RG11 4JE.

QL User Group (West Midlands): Mike Bedford-White, 16 Westfield Road, Acocks Green, Birmingham B27 7TL.

QL MUG: 3 Barnard Road, Birkenhead, L43 1TT.

SCOTLAND

Scottish QL Users Group Contact: Alan Pemberton, 65 Lingerwood Rd., Newtongrange, Midlothian EH22 4QQ. Newsletter.

HOLLAND

Sin_QL_Air (Netherlands) Membership Secretary: Bob Visser, Snelrewaard 6, 2904 SN Capelle, a/d Ijssel, Netherlands. Magazine: *Quasar*. Editor: C H M Biemans, Elzenstraat 5, 5461 CL Veghel, Netherlands.

GERMANY

Sinclair QL User Club eV (Germany) Foreign Contact: Franz Herrmann, Talstrasse 21, d-W5460 Ochenfels, West Germany. Magazine: *Quasar*.

SPAIN

Qliper Editor: Marcos Cruz, Acacias 44, 28023 Madrid, Spain. Magazine: *Qliper*.

Systematic Machine Code Programming

In part 7, Alan Bridewell gets to grips with the complexities of inputting numbers.

This series has been building a library of small chunks of assembler language code. Each chunk will do a recognisable job within a program, and be fully annotated, so that it will be clear exactly how to join it to other chunks of code to make a program, and what changes may be necessary to make the chunks fit together.

This month we shall address the problem of inputting numbers into a program. Why should this a problem? Because keyboard input does not start as a number. It starts as a string of characters. The program then has to work out if the string will convert into a valid number, and if it does not, the program must reject it (hopefully, without halting the program prematurely).

Leading spaces

This is not such a simple matter as you might imagine. For instance, the existence of leading or trailing spaces might not cause us any problems, but they may be part of the string of characters, and have to be processed. And what about decimal points? Is 0.123 a valid number, and if so, what about .123? Is 123. as valid as 123? Then we have the matter of minus signs, and, of course index notation, like 1.23E-4, which, hopefully, we can get the program to recognise as 0.000123.

For those interested in this particular problem, there has been quite a lot of correspondence in *QL World* in recent months on error-free numeric input into Basic. That will give you an idea of the complexity of the problem in Basic. In machine code, the problems become mind-boggling! Suffice to say that my programming skills are not up to dealing adequately with the problem, even with the help of Qdos routines.

The simple way out of all this is to bypass the problem by avoiding having to input numbers directly from the keyboard. We do this by getting the program to offer a default number to the user. This number can then be increased or decreased by use of the cursor keys. In this way the program always has a number which is error-free. Not only that, but by suitable programming, w can arrange that the number is always within the range of acceptable values, integer or floating point, positive or negative, and to a suitable number of significant figures of accuracy for our purpose.

A common use of numeric input is to give parameters for drawing shapes on the screen; for instance, the x and y co-ordinates of a point, or the radius of a circle. Even if the program user can input numbers correctly, it is often difficult to know exactly what sized numbers are needed to draw what is desired. It is much better to get the program to draw a default figure, and then allow the user to alter it using cursor keys.

Clearly there are some situations when numbers must be entered directly from the keyboard. For instance, spreadsheets and databases would be pretty useless without this facility. (However, if you're already in the business of writing spreadsheets, then this article is not really aimed at you!) It is surprising how many situations there are where direct keyboard input of numbers can be avoided, and in such cases it most definitely should be avoided.

Fbyte (**Listing one**) is the starting point in all this. All it does is read the keyboard and put the value of the keypress into a store called BYTE. Any key, or combination of keys that has a character code number, will result in a value in BYTE. Clearly, this could be used for detecting keypresses for any programming purpose, not just the applications considered in this article. The only point that needs to be made is that when it encounters Fbyte, the program waits for a valid keypress. This is different from Keyrow, considered in an earlier article, which can detect any combination of keys, but does not wait if no keys are being pressed.

Suitable routines

Having recorded a keypress value, we next need suitable routines to act according to the value recorded. This requires a different routine, depending on whether the numbers to be altered are integer or floating point numbers.

Updownint (**Listing two**) is a routine to increase or decrease the value stored in INT, depending on the value found in BYTE. If this is to be used with Fbyte, then the store BYTE is extra and can be deleted. It has been included because it is possible that the BYTE value used may not have come from Fbyte. It could, for instance, be the result of some program calculation, or a flag to show a particular state had been reached, requiring an integer value to be changed.

```
Listing 1

; ************************************************************************
;                                'FBYTE'
; ************************************************************************
; THIS WILL INPUT A KEYPRESS AND RETURN THE ASCII VALUE IN BYTE
;
.FBYTE          MOVE.L    (A7),A0    ; OR 4(A7),A0 OR 8(A7),A0, ETC.
                                     ; CONSOLE CHANNEL ID IN A0
                LEA.L     BYTE,A1    ; BYTE STORE ADDRESS IN A1
                MOVE.W    #$FFFF,D3  ; INFINITE TIMEOUT
                MOVEQ     #1,D0      ; IO_FBYTE IN D0
                MOVE.W    #0,D1      ; CLEAR WORD IN D1
                TRAP      #3
                MOVE.B    D1,(A1)    ; STORE BYTE
                BRA.S     NEXTBIT    ; SKIP BYTE
;
.BYTE           DC.B      0
;
; ************************************************************************
```

```
Listing 2

; ***************************************************************************
;                              'UPDOWNINT'
; ***************************************************************************
; THIS INCREASES OR DECREASES A STORED INTEGER, DEPENDING ON THE ASCII
; VALUE OF A BYTE. THE VALUE OF THIS BYTE WILL PROBABLY BE FIXED BY THE
; 'FBYTE' ROUTINE.
; IF BYTE = 208 (UP ARROW), INTEGER = INTEGER + 1
; IF BYTE = 216 (DOWN ARROW), INTEGER = INTEGER -1
; IF BYTE = 10 (ENTER), BRANCH TO <ENTER> CODE
; IF BYTE = ANYTHING ELSE, INTEGER IS UNCHANGED
;
.BYTE               DC.B      0          ; BYTE INPUT
;
.UPDOWN             LEA.L     BYTE,A1    ; BYTE ADDRESS IN A1
                    CMPI.B    #208,(A1)  ; IS BYTE = 208 ?
                    BEQ.S     UP         ; IF SO, THEN UP
                    LEA.L     BYTE,A1    ; BYTE ADDRESS IN A1
                    CMPI.B    #216,(A1)  ; IS BYTE = 216 ?
                    BEQ.S     DOWN       ; IF SO THEN DOWN
                    LEA.L     BYTE,A1    ; BYTE ADDRESS IN A1
                    CMPI.B    #10,(A1)   ; IS BYTE = 10 ?
                    BEQ.S     ENTER      ; IF SO, BRANCH TO <ENTER> CODE
                    BRA.S     NEXTBIT    ; IF NEITHER, SKIP TO NEXT BIT
.UP                 LEA.L     INT,A1     ; INTEGER ADDRESS IN A1
                    MOVE.W    (A1),D0    ; COPY A1 TO D0
                    CMPI.W    #100,D0    ; IS INT = 100 ?
                    BEQ.S     NEXTBIT    ; IF SO, DO NOTHING
                    ADDI.W    #1,(A1)    ; ADD 1 TO INTEGER
                    BRA.S     NEXTBIT    ; SKIP TO NEXT BIT
.DOWN               LEA.L     INT,A1     ; INTEGER ADDRESS IN A1
                    MOVE.W    (A1),D0    ; COPY A1 TO D0
                    CMPI.W    #-100,D0   ; IS INT = -100 ?
                    BEQ.S     NEXTBIT    ; IF SO, DO NOTHING
                    SUBI.W    #1,(A1)    ; SUBTRACT 1 FROM INTEGER
                    BRA.S     NEXTBIT    ; SKIP TO NEXT BIT
;
.INT                DC.W      10         ; INTEGER OUTPUT
; ***************************************************************************
```

The default setting of Updownint looks for three particular values in BYTE. Assuming BYTE had been set by Fbyte, 208 would indicate <UP ARROW>, 216 would indicate <DOWN ARROW>, and 10 would indicate <ENTER> had been pressed. These can be altered as required. You may, for instance, want to detect <SHIFT>, <ALT> or <CTRL> keys in combination with the cursor keys. Or suppose you were trying to adjust the x co-ordinate of a screen pixel. You would probably wish to look for <LEFT> and <RIGHT> to alter the value. The numbers for all these can be found in your *QL User Guide*.

If BYTE = 208, then INT increases by one.

If BYTE = 216, then INT decreases by one.

If BYTE = 10, then it branches to a designated routine. Any other value of BYTE is ignored.

If the value of INT is at the default upper or lower limit, then the value does not go beyond the limit.

Again, these defaults can be altered as required. You could, for example, alter INT in step of 10, or 100, or whatever. The default limit can be changed to any required values.

```
.sting 3

; ***************************************************************************
;                              'PRINTINT'
; ***************************************************************************
; THIS WILL CONVERT AN INTEGER TO ASCII AND PRINT IT OUT
;
.PRINTINT           MOVE.L    (A7),A0    ; OR 4(A7),A0 OR 8(A7),A0, ETC.
                                         ; CONSOLE CHANNEL ID IN A0
                    LEA.L     INT,A1     ; INTEGER STORE ADDRESS IN A1
                    MOVE.W    (A1),D1    ; INTEGER IN D1
                    MOVE.W    $CE,A2     ; UT_MINT VECTOR IN A2
                    JSR       (A2)
                    BRA.S     NEXTBIT    ; SKIP INTEGER
;
.INT                DC.W      10
;
; ***************************************************************************
```

Printint (**Listing three**) will convert the integer INT into an Ascii string, and print the string to a console channel, if required. It is a very simple routine needing little explanation. Basically, the Qdos vectored routine CN_ITOD to convert the integer into an Ascii string, which it then prints into the console window. If it is to be used with Updownint, the store INT is extra and is also to be used with Fbyte to monitor the changing value of INT, as it stands, it will print new numbers in the new cursor position of the window. If it is required to over-write the old value, then the cursor will need resetting each time, and also suitable blanks printed if the new number does not completely overwrite the old.

As you can see, dealing with integer is fairly simple. By comparison, dealing with floating point numbers is far more involved. It is also much slower, an integers should always be used in preference if thy are suitable.

Floating points

In part 5 (March 1992) of this series I introduced the method normally used forputting floating point number into programs. I start with two integers with floating point numbers. They are then either multiplied together, or divided one into the other, to produce a single floating point number. This can result in a large range of floating point numbers, more than enough for most applications.

Updownfp (**Listing four**) is quite a complicated routine which does for floating point numbers what Updownint does for integers. This means that a floating point number can be either increased or decreased according to the value in BYTE. However, the routine does not itself produce the floating point number, but the two integers which are to be converted to a floating point number. It also sets a flag if the required floating point number is to be negative. It does this by detecting if BYTE = 45, which is what Fbyte gives it when the minus sign is pressed. However, the most important thing about Updownfp is that it is designed to produce a limited range of numbers to a predetermined number of significant figures of accuracy. This is probably best explained by showing what the default settings do.

As printed, Updownfp will give floating point numbers with two significant figures over a range 9E-4 (ie O.OOO9) up to 100, positive or negative. It achieves this as follows: the floating point number is the result of INT1/INT2. If BYTE = 208 (<UP> key pressed) then INT1 is increased by one until it reaches 100. If it reaches 100, then it is changed to 10, and INT2 is divided by 10. If BYTE = 216 (<DOWN> key pressed) then INT1 is decreased by one until it reaches 9. If it reaches 9, then it is changed to 99 and INT2 is multiplied by 10. Let us see the effect of this in practice.

Suppose INT1 = 98 and INT2 = 10, giving the number 98/10 = 9.8. The <UP> key will result in INT1 going up to 99 and INT2 remaining 10, giving the number 99/10 = 9.9. However, if the <UP> key is pressed again,

INT1 becomes 100. When the routine detects this, it makes INT1 = 10, and divides INT2 by 10 to make it 1. This gives the number 10/1 = 10. If the <UP> key is pressed again, INT1 is increased to 11 and INT2 remaining 1, gives the number 11/1 = 11. So a sequence of successive <UP> keypresses will produce the numbers 9.8, 9.9, 10, 11, ... etc. giving two significant figures all the time. Also, since INT1 cannot go above 100, and INT2 cannot go below 1, the maximum number is 100/1 = 100.

A similar process works in the opposite direction. Suppose INT1 = 11 and INT2 = 1 giving 11/1 = 11. The <DOWN> key will result in INT1 becoming 10 and INT2 remaining 1, giving the number 10/1 = 10. However, if the <DOWN> key is pressed again, INT1 becomes 9. When the routine detects this, it makes INT1 = 99, and multiplies INT2 by 10 to make it 10. This gives the number 99/10 = 9.9. If the down key is pressed again, INT1 becomes 98 and INT2 remains the same, giving the number 98/10 = 9.8. So a sequence of successive <DOWN> keypresses will produce the numbers 11, 10, 9.9, 9.8 ... etc., giving two significant figures all the time. Also, since INT1 cannot go below 9 and INT2 cannot go above 10000, the minimum number is 9/10000 = 0.0009 or 9E-4.

INT1 and INT2 can be the integers in the parameter table in one of the routines that uses floating point numbers to print graphics (Line, Ellipse, etc.) in which case they are extra and can be deleted.

Rate of change

There are situations where this 'fixed significant figures' mode of changing numbers may not be appropriate. For instance, changing the x and y co-ordinates of the centre of a circle would not want numbers to change slowly when close to zero, and rapidly when big. The numbers would have to change at the same rate whatever they were. In this case, we can use Updownint to give INT1 and fix INT2 to some suitable value, depending on the size of numbers required.

Having obtained two integers, we now need to convert them to the required floating point number. Again, if we are using the graphic routines (Line, Ellipse, etc.) these already contain all the required code for this. However, if we are using our floating point for some other purpose, we need a routine to generate the floating point number from the two integers. Twointstofp (**Listing five**) does this job. It involves the setting up and use of an RI stack, and some addressing relative to A6. This was dealt with in some detail in part 5 (March 1992) of this series, and I would suggest you refer back to that for more information.

Essentially, what the routine does is this. It sets up an RI stack in the program's data space. It moves INT1 onto the stack and converts it to floating point. It then does the same with INT2. With two floating point numbers on the stack, it divides one into the

```
Listing 4

; ********************************************************************
;                          'UPDOWNFP'
; ********************************************************************
; THIS INCREASES OR DECREASES TWO STORED INTEGERS, DEPENDING ON THE ASCII
; VALUE OF A BYTE. THIS VALUE WILL PROBABLY BE FIXED BY THE FBYTE ROUTINE.
; THE INTEGERS, INT1 AND INT2 ARE DESIGNED TO BE CONVERTED TO F.P. NUMBERS
; AND THEN MULTIPLIED OR DIVIDED TO PRODUCE F.P. NUMBERS OVER A WIDE
; RANGE OF VALUES. THE RANGE AND SIGNIFICANT FIGURES OF THE NUMBERS CAN
; BE ALTERED BY CHANGING HOW THE TWO INTEGERS ARE AFFECTED BY THE INPUT
; BYTE.
;
; THE DEFAULT SETTINGS ARE:-
;
; FOR MULTIPLY :-
; IF BYTE = 208 (UP ARROW), INT1 = INT1 + 1
;           IF INT1 -> 100, THEN INT1 = 10 AND INT2 = INT2 * 10
; IF BYTE = 216 (DOWN ARROW), INT1 = INT1 - 1
;           IF INT1 -> 9, THEN INT1 = 99 AND INT2 = INT2 / 10
; THIS WILL GIVE 2 SIGNIFICANT FIGURES OVER A RANGE 100 TO 1E6
;
; FOR DIVIDE :-
; IF BYTE = 208 (UP ARROW), INT1 = INT1 + 1
;           IF INT1 -> 100, THEN INT1 = 10 AND INT2 = INT2 / 10
; IF BYTE = 216 (DOWN ARROW), THEN INT1 = INT1 - 1
;           IF INT1 -> 9, THEN INT1 = 99 AND INT2 = INT2 * 10
; THIS WILL GIVE 2 SIGNIFICANT FIGURES OVER A RANGE 9E-4 TO 100
;
; IF BYTE = 45 (MINUS), CHANGE SIGN
; IF BYTE = 10 (ENTER), BRANCH TO <ENTER> CODE
; IF BYTE = ANYTHING ELSE, INTEGER IS UNCHANGED
;
.BYTE               DC.B      0          ; BYTE INPUT
;
.UPDOWNFP           LEA.L     MFLAG,A1   ; MFLAG ADDRESS IN A1
                    MOVE.W    #0,(A1)    ; CLEAR MFLAG
                    LEA.L     BYTE,A1    ; BYTE ADDRESS IN A1
                    CMPI.B    #208,(A1)  ; IS BYTE = 208 ?
                    BEQ.S     UPFP       ; IF SO, THEN UP
                    LEA.L     BYTE,A1    ; BYTE ADDRESS IN A1
                    CMPI.B    #216,(A1)  ; IS BYTE = 216 ?
                    BEQ.S     DOWNFP     ; IF SO THEN DOWN
                    LEA.L     BYTE,A1    ; BYTE ADDRESS IN A1
                    CMPI.B    #45,(A1)   ; IS BYTE = 45 ?
                    BEQ.S     SFLAG      ; IF SO, SET MFLAG
                    LEA.L     BYTE,A1    ; BYTE ADDRESS IN A1
                    CMPI.B    #10,(A1)   ; IS BYTE = 10 ?
                    BEQ       ENTER      ; IF SO, BRANCH TO <ENTER> CODE
                    BRA.S     NEXTBIT    ; IF NEITHER, SKIP TO NEXT BIT
; MAKE NUMBER BIGGER. ADJUST VALUES TO SUIT.
.UPFP               LEA.L     INT1,A1    ; INT1 ADDRESS IN A1
                    LEA.L     INT2,A2    ; INT2 ADDRESS IN A2
                    MOVE.W    (A1),D0    ; INT1 IN D0
                    CMPI.W    #100,D0    ; INT1 ALREADY 100 ?
                    BEQ.S     NEXTBIT    ; IF SO LEAVE UNCHANGED
                    ADDI.W    #1,(A1)    ; ADD 1 TO INT1
                    MOVE.W    (A1),D0    ; INT1 TO D0
                    CMPI.W    #100,D0    ; IS INT1 =100 ?
                    BNE.S     NEXTBIT    ; IF NOT, NEXTBIT
                    MOVE.W    (A2),D0    ; INT2 TO D0
                    CMPI.W    #1,D0      ; IS INT2 = 1 ?
; REPLACE PREVIOUS LINE WITH NEXT LINE IF MULTIPLYING
;                   CMPI.W    #10000,D0  ; IS INT2 =10000 IF MULTIPLYING
                    BEQ.S     NEXTBIT    ; THEN LEAVE INT1,INT2 UNCHANGED
                    MOVE.W    #10,(A1)   ; ELSE, INT1 = 10
                    MOVE.W    (A2),D0    ; INT2 IN D0
                    MOVE.W    #10,D1     ; 10 IN D1
                    DIVU      D1,D0      ; INT2/10 IF DIVIDING INT1/INT2
; REPLACE PREVIOUS LINE WITH NEXT LINE IF MULTIPLYING
;                   MULU      D1,D0      ; INT2*10 IF MULTIPLYING INT1*INT2
                    MOVE.W    D0,(A2)    ; NEW D0 VALUE IN INT2
                    BRA.S     NEXTBIT    ; SKIP TO NEXT BIT
; MAKE NUMBER SMALLER. ADJUST VALUES TO SUIT.
.DOWNFP             LEA.L     INT1,A1    ; INT1 ADDRESS IN A1
                    LEA.L     INT2,A2    ; INT2 ADDRESS IN A2
                    MOVE.W    (A1),D0    ; INT1 TO D0
                    CMPI.W    #9,D0      ; INT1 ALREADY 9 ?
                    BEQ.S     NEXTBIT    ; IF SO, LEAVE UNCHANGED
                    SUBI.W    #1,(A1)    ; SUBTRACT 1 FROM INT1
                    MOVE.W    (A1),D0    ; INT1 TO D0
                    CMPI.W    #9,D0      ; IS INT1 = 9 ?
                    BNE       NEXTBIT    ; IF NOT, NEXTBIT
                    MOVE.W    (A2),D0    ; INT2 TO D0
                    CMPI.W    #10000,D0  ; IS INT2 = 10000 IF DIVIDING
; REPLACE PREVIOUS LINE WITH NEXT LINE IF MULTIPLYING
;                   CMPI.W    #1,D0      ; IS INT2 = 1 IF MULTIPLYING
                    BEQ       NEXTBIT    ; THEN LEAVE INT1,INT2 UNCHANGED
                    MOVE.W    #99,(A1)   ; ELSE, INT1 = 99
                    MOVE.W    (A2),D0    ; INT2 IN D0
                    MOVE.W    #10,D1     ; 10 IN D1
                    MULU      D1,D0      ; INT2*10 IF DIVIDING INT1/INT2
; REPLACE PREVIOUS LINE WITH NEXT LINE IF MULTIPLYING
;                   DIVU      D1,D0      ; INT2/10 IF MULTIPLYING INT1*INT2
                    MOVE.W    D0,(A2)    ; NEW D0 VALUE INTO INT2
                    BRA.S     NEXTBIT    ; BRANCH TO NEXT BIT
; SET THE FLAG FOR A MINUS F.P. NUMBER
.SFLAG              LEA.L     MFLAG,A1   ; MFLAG ADDRESS IN A1
                    MOVE.W    #1,(A1)    ; SET MFLAG
                    BRA.S     NEXTBIT    ; BRANCH TO NEXT BIT
;
.MFLAG              DC.W      0
.INT1               DC.W      100        ; ADJUST DEFAULT SETTINGS OF
.INT2               DC.W      10000      ; INTEGERS TO SUIT
; ********************************************************************
```

```
Listing 5

; ************************************************************************
;                          'TWOINTSTOFP'
; ************************************************************************
; THIS ROUTINE TAKES TWO INTEGERS, CONVERTS THEM TO F.P., THEN EITHER
; DIVIDES OR MULTIPLIES THEM TO PRODUCE AN F.P. NUMBER. THE INTEGERS MAY
; BE FIXED BY THE 'UPDOWNFP' ROUTINE FOR F.P. KEYBOARD INPUT
;
.INT1               DC.W      0
.INT2               DC.W      0
;
; PUT RI STACK 50 BELOW TOP OF DATA SPACE.
.INTSTOFP           LEA.L     -50(A5),A1          ; RI STACK ON A1
;
; MAKE ROOM FOR 1ST INTEGER
                    SUBQ.L    #2,A1       ; INTEGER = 2 BYTES
; PUT FIRST INTEGER ON RI STACK
                    MOVE.W    INT1,0(A6,A1.L)
; CONVERT TO F.P.
                    MOVEQ     #8,D0       ; #RI_FLOAT ON D0
                    MOVE.W    $11C,A3     ; RI_EXEC ON A3
                    JSR       (A3)
; REPEAT FOR 2ND INTEGER
                    SUBQ.L    #2,A1
                    MOVE.W    INT2,0(A6,A1.L)
                    MOVEQ     #8,D0
                    JSR       (A3)
; NOW DIVIDE ONE F.P. NUMBER BY THE OTHER TO LEAVE F.P. ON RI STACK.
                    MOVEQ     #$10,D0     ; #RI_DIV ON D0
;                   MOVEQ     #$0E,D0     ; #RI_MULT ON D0 (IF MULTIPLYING)
                    JSR       (A3)
; NOW CHANGE SIGN IF MFLAG IS SET
                    LEA.L     MFLAG,A2    ; MFLAG ADDRESS IN A2
                    CMPI.W    #1,(A2)     ; IS MFLAG SET ?
                    BNE.S     NEXTBIT     ; IF NOT BRANCH TO NEXT BIT
                    MOVEQ     #$14,D0     ; #RI_NEG ON D0
                    JSR       (A3)
;
; ************************************************************************
```

characters, all the time looking for an E. If it comes to a space without finding an E, then the string is not in index form, and it proceeds to strip any trailing zero before printing the string. If it finds an E it looks for a minus sign in the next character. If it finds one, it allows two more characters after the E (the minus sign and a number). If the next character is not a minus sign, that character becomes the last character allowed in the string. Whatever string it has at that stage is printed.

Limitations

Printfp seems to work well enough, and its limitations do not present many problems. Although I can see a need for it sometimes, I personally have not had cause to use it in any program I have written, except in testing Updownfp. Normally, the kind of numbers that have to be displayed on the screen are ones that can use integers. The main exception to this must be programs written to carry out large complicated computations and then display them numerically rather than graphically or any other way.

Next time we shall look at how to use these techniques to position and draw shapes on the screen in the way cad programs do. Happy coding!

other to leave one floating point number. (It also has the alternative option of multiplying if that is what is to be done.) Finally it looks at a flag which is set if the number required is negative, and changes the sign of the floating point number if the flag is set.

If we wish to print this floating point number to a console window, we can use Printfp (**Listing six**). This should be a very simple routine, almost identical to Printint, except that it uses the Qdos vectored routine CN_FTOD to convert the floating point to an Ascii string before counting the characters (CN_FTOD does not have a character count) and then printing it. However, my experiences with this Qdos routine have not been good, and it seems to throw up trailing zeros and other spurious digits after the string. Since it also does not tell us how long the string is, finding its true end can be a bit difficult. I don't know whether all this is caused by a flaw in my programming, or a Qdos bug, or what, and I would be pleased to hear from anyone who can throw light on the matter.

Single digits only

To get over these problems, I have arranged for Printfp to handle only single digit exponents (ie in the range E-9 to E9). This will give a range of numbers big enough for most purposes.

First the routine sorts out the row and column position of the cursor for printing. It then uses CN_FTOD to obtain the Ascii string for the floating point number. It next proceeds to work its way through the string counting the

```
Listing 6
; ************************************************************************
;                          'PRINTFP'
; ************************************************************************
; THIS WILL PRINT OUT AN F.P. NUMBER ON THE TOP OF THE RI STACK AT A GIVEN
; POSITION IN THE CONSOLE WINDOW.
; ** NOTE ** BECAUSE IT TRUNCATES THE STRING TO REMOVE TRAILING ZEROS AND
; SPURIOUS DIGITS, IT WILL ONLY ALLOW SINGLE DIGIT EXPONENTS (+ OR -),
; GIVING EXPONENTS IN THE RANGE E-9 TO E9.
;
; MOVE CURSOR TO PRINT POSITION
.PRINTFP            MOVE.L    (A7),A0     ; OR 4(A7),A0 OR 8(A7),A0 ETC.
                                          ; CONSOLE CHANNEL ID IN A0
                    MOVEQ     #$10,D0     ; #SD_POS IN D0
                    MOVE.W    #$0,D1      ; COLUMN 0 ADJUST TO REQUIREMENTS
                    MOVE.W    #$0,D2      ; ROW 0 ADJUST TO REQUIREMENTS
                    MOVE.W    #$FFFF,D3   ; INFINITE TIMEOUT
                    TRAP      #3          ;
; CONVERT F.P. NUMBER ON RI STACK TO ASCII STRING
                    LEA.L     STRING,A0   ; STRING ADDRESS IN A0
                    SUBA.L    A6,A0       ; MAKE A0 RELATIVE TO A6
                    MOVE.W    #F0,A3      ; CN_FTOD IN A3
                    JSR       (A3)
; GET STRING LENGTH MINUS TRAILING ZEROS AND/OR SPURIOUS DIGITS
                    LEA.L     STRING,A0   ; STRING ADDRESS IN A0
                    LEA.L     STRING_L,A1 ; STRING_L ADDRESS IN A1
                    MOVE.W    #0,(A1)     ; SET STRING LENGTH TO ZERO
.NEXTCHR            MOVE.B    (A0)+,D0    ; PUT NEXT CHR IN D0
                    MOVE.B    D0,D1       ; COPY CHR INTO D1
                    CMPI.B    #32,D0      ; IS CHR A SPACE ?
                    BEQ.S     PREVCHR     ; IF SO, CLEAR ANY TRAILING ZEROS
                    ADDI.W    #1,(A1)     ; IF NOT, INCREMENT STRING LENGTH
                    CMPI.B    #69,D1      ; IS CHR 'E' ?
                    BEQ.S     MINUS       ; IF SO, LOOK FOR MINUS SIGN
                    BRA.S     NEXTCHR     ; IF NOT, LOOP TO NEXT CHR
; IF 'E' FOR EXPONENT IS DETECTED, THE NEXT BIT LOOKS FOR MINUS SIGN.
; IF ONE IS FOUND, THERE ARE TWO MORE CHARACTERS, OTHERWISE ONLY ONE.
.MINUS              MOVE.B    (A0)+,D0    ; PUT NEXT CHR IN D0
                    CMPI.B    #45,D0      ; DOES MINUS SIGN FOLLOW 'E' ?
                    BNE.S     ONEMORE     ; IF NOT, ONLY ONE MORE CHR
                    ADDI.W    #2,(A1)     ; ADD 2 TO STRING LENGTH
                    BRA.S     PSTRING     ; PRINT TRUNCATED STRING
.ONEMORE            ADDI.W    #1,(A1)     ; ADD ONE TO STRING LENGTH
                    BRA.S     PSTRING     ; PRINT TRUNCATED STRING
; THE NEXT BIT REMOVES TRAILING ZEROS IF THERE IS NO EXPONENT.
.PREVCHR            SUBA.L    #1,A0       ; GET PREVIOUS CHR
                    MOVE.B    (A0),D0     ; PUT CHR IN D0
                    MOVE.B    D0,D1       ; AND COPY INTO D1
                    CMPI.B    #32,D0      ; IS IT A SPACE ?
                    BEQ.S     PREVCHR     ; IF SO, LOOP TO PREVIOUS CHR
                    CMPI.B    #48,D1      ; IS IT ZERO ?
                    BNE.S     PSTRING     ; IF NOT PRINT STRING
                    SUBI.W    #1,(A1)     ; REDUCE STRING LENGTH BY ONE
                    BRA.S     PREVCHR     ; LOOP TO PREVIOUS CHR
; PRINT STRING
.PSTRING            MOVE.L    (A7),A0     ;CHANNEL ID IN A0
                    MOVE.W    #D0,A2      ;UT_MTEXT IN A2
                    JSR       (A2)
                    BRA       FBYTE
;
.STRING_L           DC.W      10          ; ENOUGH SPACE FOR
.STRING             DC.B      '          ' ; ANY F.P. STRING
; ************************************************************************
```

DJC

Dilwyn Jones Computing

41 Bro Emrys, Tal-y-Bont,
Bangor, Gwynedd LL57 3YT U.K.
Tel: Bangor (0248) 354023

GRAPHICS

THE PAINTER V4.04 ..... **£25.00**
[F 512K] 100% machine code art and graphics program, by PROGS of Belgium. Multiple screens, can be mouse controlled, selection of fonts, patterns, many facilities.

THE CLIPART **£12.00**
[F 128K] 3 disks full of clipart pictures, for use with most DTP/graphics programs.

QRACTAL **£20.00**
[F 512K] Machine code fractals program. Displays Julia or Mandelbrot sets.

QRACTAL SCREENS ..... **£ 5.00**
[F 128K] A disk full of sample screen pictures generated with the QRactal program. Buy this to see the sample pictures, then ask about a discount when buying QRactal itself!

IMAGE PROCESSOR V2 ..... **£15.00**
[F 256K] Image enhancement, edge detection, zoom and edit, touch up picture generated by digitisers, scanners or other graphics programs. Works with mode 4 or 8 screens.
UPGRADE OLD IMAGE PROCESSOR **£10.00**
(Send proof of purchase of old version)

PD2 CLIPART **£10.00**
[F 128K] Clipart made for PD2 Plus, but available in screen format for use with other DTP/graphics programs if required.

SCREEN SNATCHER **£10.00**
[F IM 128K] 'Grab' screen displays from within other programs that are not able to save their own displays.

TEXT 'N' GRAPHIX **£20.00**
[F 256K] **NEW!** Allows you to insert screens, or part of QL screen pictures, into a Quill or plain text file printout. Mix NLQ text and graphics in the same printout! Currently only for use with 9 pin Epson compatible dot matrix printers - ask about a version for 24 pin printers due soon.

TRANS24 ..... **£10.00**
[F IM 128K] Translates 9 pin graphics printouts into data for a 24 pin printer.

TEXT

BIBLE TEXT DISKS, EDITOR FORMAT **£20.00**
BIBLE TEXT DISKS, QUILL "_DOC" **£20.00**
[F 256K] Text of the King James Bible on Disk. Please state whether you require Editor (plain text) or Quill '_doc' format.

SPELLBOUND **£30.00**
(F 1M 384K) A spelling checker which can check spelling *as you type!* 30,000 word dictionary

SPELLBOUND SPECIAL EDITION **£50.00**
(F 512K) Enganced version of Spellbound with 50.000 word dictionary & many new features.
UPGRADE TO SPELLBOUND S.E. **£30.00**

QUICK POSTERS **£10.00**
(F 2M 128K) Text poster maker, for use with Star NL, XB and LC printers.

ROB ROY BARGAIN PACK **£10.00**
(F 3M 128K) Reviewed in QL WoRld August 1991

DATA BASES

ADDRESS BOOK & LABEL PRINTER **£15.00**
[F 2M R 384K] Store names and addresses and print them out on a variety of label sizes, or print a telephone list, etc.

DATA DESIGN **£50.00**
[F 512K] Superb, fast 100% machine code database by PROGS of Belgium. Programmable via BASIC and machine code interfaces - write your own database applications in BASIC using the DATA Design commands to help you! Runs in pointer environment (supplied)

FLASHBACK **£25.00**
[F 1M 256K] Fast machine code database which is also very easy to use.

FLASHBACK SPECIAL EDITION **£40.00**
[F 2M 512K] Enhanced version of Flashback, new commands, report generator, etc
UPGRADE TO FLASHBACK S.E. **£20.00**
(return Flashback master disk)

QL GENEALOGIST, STANDARD VERSION ... **£19.50**
[F 2M 384K] Family trees and family history database, one of our best selling programs.

QL GENEALOGIST SECOND EDITION **£30.00**
[F 384K] Enhanced version - for details see our advert in QL World December 1991.
UPGRADE TO SECOND EDITION **£12.00**
(return Genealogist master disk)

BUDGET 128K QL GENEALOGIST **£12.00**
[F IM 128K] Cut down version for unexpanded machines.

SUPER DISK INDEXER **£12.00**
[F IM 384K] Catalogue your disk collection as a database of files. Store it, print it, search & find files, make your life easier.

DBEASY **£15.00**
[F 512K] *NEW!* A database front end for Archive, plus a suite of programs called Chaos Busters. Software from EMSoft, U.S.A.

DBPROGS **£15.00**
[F 512K] *NEW!* A collection of Archive utilities and text files to help you learn to program Archive. Software from EMSoft, U.S.A.

DTP

PAGE DESIGNER 2 PLUS **£40.00**
[F 512K] QL Desktop Publishing program, mix text and graphics, make printed pages or posters, use clipart, use graphics from other programs etc. Prints on most printers. Ask for details! (please note: this program has been delayed, apologies to those waiting– please check if available before ordering)

UPGRADE PD2 TO PD2 PLUS **£20.00**
(Send proof of purchase of old PD2)

QL HARDWARE

MINI PROCESS CONTROLLER **£59.95**
Relay switched outputs, controlled via QQL serial port.

SOFTWARE TOOLKIT FOR MPC ABOVE **£9.95**
[F 128K] Control the MPC from BASIC programs more easily with these extensions.

NETWORK PROVER ..... **£ 3.50**
Plugs between network lead and computer to give a visual indication of transmission.

POSTAGE RATES - SEE BELOW

OTHER QL SOFTWARE

PRINTERMASTER ..... **£20.00.**
[F IM 128K] *NEW!* Tame your printer! Popup utility with menu selection of control codes to send to a printer, Want to change fonts, set paper length, margins, set up for listings or wordprocessing etc. without having to refer to the manual and struggle with control codes. Let Printermaster take the strain out of living with your printer.

HOME BUDGET **£20.00**
[F IM 128K] Domestic bills and accounts program, plus a UK Income Tax Calculator.

REMIND-ME **£12.00**
[F IM 128K] Dates and events reminder program - remember about birthdays, licence renewals, etc. Quick and very easy to use.

REMIND-ME PLUS **£20.00**
(F1M 128k) NEW! Enhanced version with ability to schedule twice as many dates, longer event descriptions, etc.
UPGRADE REMIND-ME TO REMIND-ME PLUS ... **£10.00**
Return master disk with order for upgrade

SCREEN ECONOMISER **£10.00**
[F IM 128K] Turns off the QL display after a set number of minutes to protect the screen.

SLOWGOLD ..... **£ 5.00**
[F 128K] Slowdown routine and control panel for software which runs too fast on Gold Card or indeed any QL system.

TASKMASTER **£25.00**
[F IM 384K] Task switching utility. Enables you to conveniently switch between several programs in memory. Calculator, notepad and file handling utility included.

SUPER DISK LABELLER **£10.00**
[F 256K] Print neat labels for your floppy disks listing the filenames on the disk in columns in small print.

THE CAT ..... **£ 5.00**
[F IM 128K] List files on a disk or cartridge to the screen, or print on paper, in columns (optionally sorted in this latest version). Useful and convenient utility.

★ SEE ALSO THE OTHER HALF OF THIS ADVERT ON THE PRECEDING PAGE ★

SOFTWARE POSTAGE: Software sent post-free to UK addresses. Abroad add £1.00 per program for postage and packing.
SUPPLIES POSTAGE: For disks, boxes and stands, add £2.50 for postage to UK addresses, or 10% of order value (minimum postage £3.50) for airmail postage where possible.
PAYMENT:We can accept payment by cash, cheque (in UK Pounds Sterling only, drawn on UK branch of bank or building society), Postal Order, International Postal/Money order, or Eurocheque. We can also accept payment by these credit cards: VISA, ACCESS, MASTERCARD, EUROCARD and also Barclays CONNECT card. Please state your card type, number, expiry date and your address (goods paid for by credit card can only be sent to cardholder's address). Remember to sign your order. Goods remain property of DJC until paid for in full. Our telephone number above has an answering machine for when we are unable to answer in person – don't be shy!

VISA · Access · EUROCARD · Master Card